MUNIX

PROGRAMMER'S MANUAL

VOLUME 2C

PCS GmbH
Periphere Computer Systeme
Pfälzer-Wald-Straße 36
8000 München 90
Telefon (0 89) 68 10 21
Telex 5 23 271

MUNIX Programmer's Manual

Volume 2C

This volume contains some documents which are useful during the first days of working with MUNIX, and others which give further information about extensions and differences to UNIX* V7.

1. Setting up MUNIX on a QU68000

   Recipes to bring up the MUNIX system for the first time.

2. MUNIX - Erste Schritte auf dem QU68000

   Einfuehrender Artikel fuer den Systemverwalter.

3. MUNIX - Summary

   Software facilities available on MUNIX.

4. Deviations of MUNIX/M from MUNIX

5. Unterschiede von MUNIX zu UNIX V7

   Abweichungen der Implementierung von UNIX V7 auf dem QU68000 gegenueber der PDP11 Version von UNIX V7.

6. Assembler 68000 User's Guide

7. FSCK - The UNIX File System Check Program

   Describes the interactive filesystem check and repair program.

8. Berkeley Font Catalog

   Samples of various fonts available using vtroff on a Versatec printer.

---

*UNIX is a Trademark of Bell Laboratories.

October 29, 1982

Setting up MUNIX on a QU68000

PCS
Pfaelzer-Wald-Str. 36
8000 Muenchen 90

This paper contains a set of recipes how to bring up your MUNIX system on different hardware configurations.

Choose the appropriate description:

Your MUNIX/M is configured for a RL02 disk and delivered on...

mini-floppies (5 1/4") i.e. for QU68000-CWF 15/1	description no. 1
floppies (8")	no. 2

Your MUNIX is always delivered on tape and configured for...

RK06/07 (80MB Fujitsu)	description no. 3
RL02	no. 4

If MUNIX has already being set up by PCS, choose the appropriate description and start with 'Start MUNIX'.

October 29, 1982

DESCRIPTION NO. 1

Setting up MUNIX on a QU68000-CWF 15/1

Your copy of MUNIX is delivered on 12 double sided/ double density mini-floppies. You got:

1) one floppy with standalone programs to set up MUNIX

2) six floppies with the root filesystem dump

3) five floppies with the /usr filesystem dump

The 14,4 MByte Winchester-Disk emulates a RL02 disk.

MUNIX is configured for RL02, unit 0 as the root, swap and pipe device.

One physical RL02 unit is divided into two logical halves with max. 10240 512 byte blocks each.

DEC-name	MUNIX special files	
DL0	/dev/rl0	/dev/rrl0
	/dev/rl1	/dev/rrl1
DL1	/dev/rl2	/dev/rrl2
	/dev/rl3	/dev/rrl3
		raw devices

and so on.

rl0 is the root device. It comprises 8000 blocks. The rest of the first logical half of DL0 is reserved for the swap area.

rl1 comprises 10240 blocks.

The procedure to set up MUNIX is as follows:

0) Power on

1) Bus-reset or init
The Minitor (boot PROM) prompts with:
.Minitor

2) Insert the floppy with standalone programs into drive 0.

3) Load the root filesystem:

Enter (system messages are underlined):

```
.rx                          (boot from floppy)
./boot                       (load boot program)
.g0                          (start)
     ...
start of c 68000
Boot
: rx(2,0)mkfs                (load mkfs from floppy
                              and start)
file system size: 8000       (root filesystem size)
file system: r1(0,0)
Exit called
Boot
: rx(2,0)restor              (load restor from floppy
                              and start)
tape? rx(2,0)                (before entering this line insert
                              root dump floppy no.1 into drive 0)
disk? r1(0,0)                (root filesystem from floppies to disk)
last chance before...        (press "return")
...
mount volume 2               (insert root dump floppy no.2 (drive 0)
...                           and press return)
...                          (and so on)
end of tape
```

4) For loading the /usr filesystem please repeat steps 1) to 3) with the following arguments:

```
mkfs:    file system size = 10240,
         file system = r1(0,10240)

restor: (insert the 1. /usr dump floppy)
         tape? rx(2,0)
         disk? r1(0,10240)
```

5) Start MUNIX:

```
Bus-reset or init:
.Minitor
.rl
./unix
.g0
     ...
Start of c 68000
     ...
Start MUNIX (V1.3) on QU68000
```

6) MUNIX is now running.

7) One of the first steps you should do is to check the installed filesystems by /etc/fsck.

Good luck with your MUNIX/QU68000 system!

DESCRIPTION NO. 2

Setting up MUNIX on a QU68000

Your copy of MUNIX is delivered on 7 double sided/ double density floppies. You got:

1) one floppy with standalone programs to set up MUNIX

2) three floppies with the root filesystem dump

3) three floppies with the /usr filesystem dump

MUNIX is configured for RL02, unit 0 as the root, swap and pipe device.

One physical RL02 unit is divided into two logical halves with max. 10240 512 byte blocks each.

DEC-name	MUNIX special files	
DL0	/dev/rl0	/dev/rrl0
	/dev/rl1	/dev/rrl1
DL1	/dev/rl2	/dev/rrl2
	/dev/rl3	/dev/rrl3
		raw devices

and so on.

rl0 is the root device. It comprises 8000 blocks. The rest of the first logical half of DL0 is reserved for the swap area.

rl1 comprises 10240 blocks.

The procedure to set up MUNIX is as follows:

0) Power on

1) Bus-reset or init
The Minitor (boot PROM) prompts with:
.Minitor

2) Insert the floppy with standalone programs into drive 0.

3) Load the root filesystem:

Enter (system messages are underlined):

```
.rx                          (boot from floppy)
./boot                       (load boot program)
.g0                          (start)
     ...
start of c 68000
Boot
: rx(2,0)mkfs                (load mkfs from floppy
                              and start)
file system size: 8000 (root filesystem size)
file system: rl(0,0)
Exit called
Boot
: rx(2,0)restor              (load restor from floppy
                              and start)
tape? rx(2,0)                (before entering this line insert
                              root dump floppy no.1 into drive 0)
disk? rl(0,0)                (root filesystem from floppies to disk)
last chance before...        (press "return")
...
mount volume 2               (insert root dump floppy no.2 (drive 0)
...                           and press return)
...                          (and so on)
end of tape
```

4) For loading the /usr filesystem please repeat steps 1) to 3) with the following arguments:

```
mkfs:   file system size = 10240,
        file system = rl(0,10240)

restor: (insert the 1. /usr dump floppy)
        tape? rx(2,0)
        disk? rl(0,10240)
```

5) Start MUNIX:

```
Bus-reset or init:
.Minitor
.r1
./unix
.g0
     ...
Start of c 68000
     ...
Start MUNIX (V1.3) on QU68000
```

6) MUNIX is now running.

7) One of the first steps you should do is to check the installed filesystems by /etc/fsck.

Good luck with your MUNIX/QU68000 system!

DESCRIPTION NO. 3

Setting up MUNIX on a 0U68000

Your copy of MUNIX is delivered on a 9 track 1600 bpi tape. It contains following files:

0) standalone boot program

1) standalone mkfs (make file system)

2) standalone restor (restore dumps)

3) root filesystem dump

4) /usr filesystem in tar format

Your 80 MB Fujitsu emulates two RK07 disks and one RK06. MUNIX is configured for the first RK07 as root, swap and pipe device. The second RK07 is reserved for the /usr filesystem. The physical drives can be divided into several logical devices. For further details of the logical organization of the Fujitsu see HK(4) in MUNIX Vol. 1.

drive	MUNIX special files	filesystem size	use
0: RK07	/dev/hk0	9636	root,pipe
	/dev/swap	8910	swap
	/dev/hk1	35178	free
1: RK07	/dev/hk2	53724	usr
2: RK06	/dev/hk3	27060	free

Procedure to set up MUNIX:

0) Change the set of EPROMS on the processor board for booting from tape (R900.055, Min 1.1).

1) Power on

2) Bus-reset or init
The Minitor (boot PROM) prompts with:
.Minitor

3) Mount the tape.

4) Load the root filesystem:

Enter (system messages are underlined):

```
.rt                          (boot from tape)
./boot                       (load boot program)
.g0                          (start)
 ...
start of c 68000
Boot
: tm(0,1)                    (load mkfs from tape
                              and start)
file system size: 9636 (root filesystem size)
file system: hk(0,0)
Exit called
Boot
: tm(0,2)                    (load restor from tape
                              and start)
tape? tm(0,3)                (root filesystem from tape
disk? hk(0,0)                 to disk)
last chance before...        (press "return")
end of tape
```

5) Change the set of EPROMS for booting from the Fujitsu disk (R900.056, Min 1.2).

6) Start MUNIX:

```
BUS-reset or init:
.Minitor
./unix                    (load from hk is default)
.g0
 ...
Start of c 68000
 ...
Start MUNIX (V1.3) on QU68000
```

7) MUNIX is now running.

8) The /usr filesystem is divided into two tar-files. The first contents a 'mini-usr-filesystem', the second additional facilities like computer-aided instruction, on-line manuals and support for the Versatec printer.

After creating a filesystem on the hk2, mount hk2 on /usr. Skip the first five files of the tape. (Five, because between the dump-file and the tar-file a very small file is created.) Take the root as current directory (cd /) and finally extract the mini-usr-filesystem by typing:

```
tar xvbf 20 /dev/rmt0
```

Extract the second tar-file accordingly.

9) One of the first steps you should do is to check the installed filesystems by /etc/fsck.

Good luck with your MUNIX/QU68000 system!

DESCRIPTION NO. 4

Setting up MUNIX on a QU68000

Your copy of MUNIX is delivered on a 9 track 1600 bpi tape. It contains the following files:

0) standalone boot program

1) standalone mkfs (make file system)

2) standalone restor (restore dumps)

3) root filesystem dump

4) /usr filesystem in tar format

MUNIX is configured for RL02, unit 0 as the root, swap and pipe device.

One physical RL02 unit is divided into two logical halves with max. 10240 512 byte blocks each.

DEC-name	MUNIX special files	
DL0	/dev/rl0	/dev/rrl0
	/dev/rl1	/dev/rrl1
DL1	/dev/rl2	/dev/rrl2
	/dev/rl3	/dev/rrl3
		raw devices

and so on.

rl0 is the root device. It comprises 8000 blocks. The rest of the first logical half of DL0 is reserved for the swap area.

rl1 comprises 10240 blocks.

The procedure to set up MUNIX is as follows:

0) Power on

1) Bus-reset or init
The Minitor (boot PROM) prompts with:
.Minitor

2) Mount the tape.

3) Load the root filesystem:

Enter (system messages are underlined):

```
.rt                          (boot from tape)
./boot                       (load boot program)
.g0                          (start)
     ...
start of c 68000
Boot
: tm(0,1)                    (load mkfs from tape
                              and start)
file system size: 8000       (root filesystem size)
file system: rl(0,0)
Exit called
Boot
: tm(0,2)                    (load restor from tape
                              and start)
tape? tm(0,3)                (root filesystem from
disk? rl(0,0)                 tape to disk)
last chance before...        (press "return")
end of tape
```

4) Start MUNIX:

```
Bus-reset or init:
.Minitor
.rl
./unix
.g0
     ...
Start of c 68000
     ...
Start MUNIX (V1.3) on QU68000
```

5) MUNIX is now running.

6) The quantity of the /usr filesystem is more than a filesystem on a RL02 can take.
The /usr filesystem is therefore divided into two tar-files. The first contents a 'mini-usr-filesystem' we normally deliver for a configuration with a RL02. The second contents additional facilities like computer aided instruction, on line manuals and support for a raster printer/plotter.
We suppose to use only the mini-usr-filesystem and extract those parts of the second tar-file you really need.

After creating a filesystem on the rl1, mount rl1 on /usr. Skip the first five files of the tape. (Five, because between the dump-file and the tar-file a very small file is created.) Take the root as current directory (cd /) and finally extract the mini-usr-filesystem by typing:

```
tar xvbf 20 /dev/rmt0
```

7) One of the first steps you should do is to check the installed filesystems by /etc/fsck.

Good luck with your MUNIX/QU68000 system!

Munix - Erste Schritte auf dem QU68000

ABSTRACT

Im folgenden werden die ersten Schritte beschrieben, die Sie machen müssen, um Unix auf Ihrer Anlage zum Laufen zu bringen.

Minitor-Grundlagen

Legen Sie alle 3 Schalter am Rechner nach oben und schalten Sie das Netz ein. Auf der Konsole muss ein Punkt als Minitor-Prompt erscheinen. Erscheint der Punkt auch nach Betätigen der Reset-Taste nicht, überprüfen Sie die CPU, den Speicher und den Anschluss der Konsole. Mit dem Minitor können Sie Programme von drei unterschiedlichen Geräten booten, nämlich der RX02 Floppy, der RL02 Platte und der RK06/07 Platte. Mehr Treiber lassen sich zur Zeit aus Platzgründen im Minitor nicht unterbringen, da ja auch noch Debugging-Hilfen Teil des Minitors sind. Sie haben eine Unix-Boot-Floppy bekommen, und entweder eine fertig bespielte Platte, oder ein Magnetband. Die folgende Beschreibung der Konfigurierung geht davon aus, dass auf Ihrer Platte, dem root-device, schon ein root-Filesystem vorhanden ist.

Versuchen sie erst, Unix von dieser Platte zu laden, indem Sie einfach

rl <return> oder r0<return>, und dann
/aunix <return>

eintippen. Kommt eine Fehlermeldung, kann der Minitor von dieser Platte nicht booten, und Sie müssen von der Floppy booten:

rx <return>
/aunix <return>

Eine Minitor-Version kann auch von Band booten (mit "rt"). Nach /aunix vergeht etwas Zeit, bis Unix geladen ist, dann meldet sich wieder der Minitor mit ".". Starten Sie nun Unix mit

g <return>

Es muss die Meldung "Start of C 68000" kommen. Anschliessend führt Unix einen Dialog mit Ihnen, der dazu dient, auf Ihrer Anlage ein erstes Unix zum Laufen zu bringen, mit dessen Hilfe dann die eigentliche Generierung erfolgt.

Root device

Zuerst wird nach dem root-device gefragt. Geben sie einen von den Gerätenamen an (HK steht dabei für RK06/RK07). Anschliessend wird nach der minor device number gefragt, d.h. nach der Nummer des logischen Gerätes. Diese ist nur manchmal mit der physikalischen unit number identisch. Im einzelnen:

rx Nur RX02 wird unterstützt. Es sind zwei physikalische Laufwerke vorgesehen. Unit 0 und 2 sind Floppy 0 (linkes Laufwerk), unit 1 und 3 Floppy 1 (rechts). 0 und 1 sind single density, 2 und 3 sind double density. Die Anzahl der Blöcke pro Floppy sind 500 (single density, single sided), 1001 (single density, double sided oder double density, single sided) oder 2002 (double density, double sided). Die Blockanzahl ist wichtig für das Programm mkfs.

rk Für die maximal vier Laufwerke ist physikalische gleich logischer Nummer. Die RK05 Platte ist 4872 Blöcke gross.

rl Die RL02-Platte wird logisch in zwei Hälften unterteilt. Haben Sie zwei RL02-Laufwerke, sind die logischen units 0 und 1 auf der physikalischen unit 0, und 2 und 3 auf unit 1. Die logischen units sind je 10240 Blöcke gross.

hk Die minor device numbers der RK06/07 Platten berechnen sich als physical unit number * 8 + logical unit number. Die RK06/07 Platten lassen sich nach folgender Tabelle in logische units zerlegen:

unit	size	cyloff	
0,	9636,	0,	cyl 0-145, root on rk06/7
1,	8910,	146,	cyl 146-280, swap on rk06/7
2,	8514,	281,	cyl 281-409, rest of rk06 pack
3,	35178,	281,	cyl 281-813, rest of rk07 pack
4,	0,	0,	spare, for avoiding bad blocks
5,	0,	0,	spare, for avoiding bad blocks
6,	27060,	0,	cyl 0-409, all of rk06 pack
7,	53724,	0,	cyl 0-813, all of rk07 pack

Wir haben beispielsweise eine Fujitsu 2312 Platte, 80 Megabyte unformatiert, die durch einen Emulex SC02 controller als zwei RK07 und eine RK06 emuliert wird. Diese drei physikalischen units werden bei uns in 5 logische units aufgeteilt:

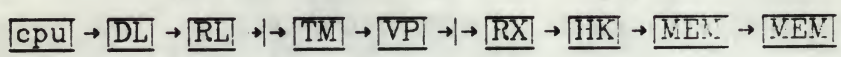
hk0: minor device 0 = 0 * 8 + 0,
d.h. die ersten 9636 Blöcke der ersten RK07, dient als root-system
swap: minor device 1 = 0 * 8 + 1,
d.h. die nächsten 8910 Blöcke der ersten RK07, dient als swap device
hk1: minor device 3 = 0 * 8 + 3,
d.h. der Rest der ersten RK07
hk2: minor device 15 = 1 * 8 + 7,
d.h. die ganze zweite RK07
hk3: minor device 22 = 2 * 8 + 6,
d.h. die ganze RK06

Swap device

Nachdem das root-device spezifiziert wurde, wird nach dem swap-device gefragt. Wieder muss Plattentyp und minor device number angegeben werden. Weiterhin wird nach swplo, der Nummer des ersten Blocks des swap-Bereichs, und nswap, der Länge des swap-Bereichs, gefragt. Bei HK besteht die Möglichkeit, eine eigene logische unit als swap-device anzugeben. Bei den anderen Platten muss eine logische unit aufgeteilt werden, so dass teilweise ein Filesystem und ein swap-Bereich auf der Platte untergebracht werden. Beispiel: Ich habe zwei RK05, von denen die unit 0 eine Wechselplatte und die unit 1 eine Festplatte ist. Ich möchte root und swap auf rk1 legen. Der swap-Bereich muss mindestens 1000 Blöcke gross sein. Ich werde dann beim Dialog für root- und swap-device beide Male rk und 1 eingeben, bei swplo 3500 und bei nswap 1372 (4872 - 3500). Damit kann ich auf rk1 mit mkfs ein filesystem der Länge 3500 einrichten, an das anschliessend der swap-Bereich mit Länge 1372 folgt. Wenn ich versehentlich (alles schon passiert) auf rk1 ein Filesystem der Länge 4872 einrichte, werden mir, wenn das System zu swappen anfängt, alle Blöcke grösser als 3500 überschrieben.

DMA-Extension-Register

Die nächste Frage bezieht sich auf die DMA-Extension-Register (DER). Wenn auf der Rückwand Ihres Rechners noch keine DER-Verdrahtung ist, muss immer 0 angegeben werden. Die DER-Verdrahtung ordnet jedem Gerät ein DER zu. Das geschieht durch zwei Anzapfungen der Daisychain. Die Daisychain verbindet den Ausgang BSACKO einer Karte mit dem Eingang BSACKI der nachfolgenden Karte. Wenn der Prozessor nicht Bus-Master ist, aktiviert er sein BSACKO. Jede Karte, die nicht Busmaster ist, gibt BSACKI an BSACKO weiter. Der Busmaster hingegen hält BSACKO auf 0. Bus-Master ist also die Karte, für die BSACKI = 1 und BSACKO = 0 ist. Mit den zwei Anzapfungen der Daisy-chain können die Werte 00, 10 oder 11 gewonnen werden, je nachdem ob eine Karte Bus-Master ist, die vor dem ersten Anzapf, zwischen erstem und zweitem, oder nach dem zweiten Anzapf liegt. Im Beispiel



wird so dem Gerät RL das DER 0, TM und VP das DER 2 und RX und HK das DER 3 zugeordnet. Für Karten, die nicht Bus-Master werden können, wie DLV11 oder Speicher, ist die Stellung in der Daisy-chain beliebig.

Die Reihenfolge der Karten im Q-Bus bestimmt ausserdem die Hardware-Priorität. Bei der Plazierung der Karten sollte man von folgenden Richtlinien ausgehen:

a) Terminals nach vorne.

b) Geräte mit 22-bit DMA-controller auf DER 0. Wenn solche Geräte vorhanden sind, dann nur diese auf DER 0.

c) Ansonsten DMA-Geräte gleichmässig auf 0 (wenn ohne 22bit DMA), 2 und 3 verteilen.

d) Lücken mit Speicherkarten füllen. Wenn spätere Erweiterung, Speicher nach hinten und neues Gerät in die Lücke.

Non-Standard Geräte

Die letzten zwei Fragen beruhen auf Erfahrungen, die wir mit Geräten gesammelt haben, die sich nicht an die Q-Bus Spezifikationen halten. So kann es sein, dass ein Gerät eine Q-Bus-Addresse als in der IO-Page liegend ansieht, wenn die Addressbits 15, 16 und 17 = 1 sind, obwohl das Signal BBS7 nicht aktiv ist. Das führt dazu, dass ein Ansprechen der oberen 16 kbyte jedes 256kbyte-Speicherbereichs dieses Gerät manchmal Amok laufen lässt. Ein anderer Fehler ist, dass ein Gerät zu lange den Bus blockiert, indem es z.B. 256 Worte in einem Schub überträgt. Andere Geräte, die sich an die Spezifikation halten, können dann Daten verlieren oder Fehler produzieren. Wenn Sie die Frage, die sich auf den ersten Fehler bezieht, bejahen, lässt das System jeweils die oberen 16kb frei. Bejahen Sie die zweite Frage, wird dafür gesorgt, dass nur jeweils ein DMA-Gerät zur Zeit aktiv sein kann (Effizienzverlust!).

Hochfahren

Nach der letzten Frage wird Unix im single-user-mode hochgefahren. Auf der Konsole erscheint ein "#" als prompt. Tippen Sie nun das Datum ein (siehe Date I/1). Gehen Sie nach /etc (cd /etc) und beschauen sie sich (cat) die Dateien rc und ttys. rc ist ein Shell-File, das jedesmal beim Übergang vom single-user-mode zum multi-user-mode ausgeführt wird. In ttys muss Ihre Terminal-Konfiguration erstellt werden. Das erste Zeichen der Zeile muss 0 oder 1 sein, je nachdem, ob für dieses Terminal keine oder eine Shell initiiert wird. Das zweite Zeichen gibt die Art des Terminals an (z.B. o für Perkin Elmer Owl, t für Tandberg TDV2215, v für vt100, p für Siemens PT80). Ist Ihr Terminal keins von diesen, wird lediglich die Clear-screen-Sequenz vor dem login nicht richtig funktionieren. Beachten Sie auch die Dateien passwd, fstab, ttytype und termcap. Die Beschreibung dieser Dateien finden Sie in I/5. Gehen Sie nun in das directory /dev. Löschen Sie die special files, die Sie nicht benötigen. Fahren Sie nun in den Multi-user-mode hoch, indem Sie Ctrl-Z drücken. An den eingeschalteten und in ttys generierten Terminals muss "login" erscheinen.

Terminal-Konventionen

Anders als in der Unix-Dokumentation beschrieben, können Sie das zuletzt eingegebene Zeichen mit BS oder DEL löschen (anstatt #), und die ganze Zeile mit Ctrl-X (anstatt @). Das EOT-Zeichen ist nicht Ctrl-D, sondern Ctrl-Z. Zum Unterbrechen dienen Ctrl-C (SIGINTR) und Ctrl-Y (SIGQUIT). Diese Konventionen stammen von VAX-VMS.

Editoren

An Zeileneditoren laufen natürlich ed und sed. An Bildschirmeditoren gibt es vorerst ped (von uns selbst), und vi (von Berkeley). Die Bildschirmeditoren passen sich unterschiedlichen Terminals an. In /etc/termcap soll eine Liste der an Ihrer Installation vorhandenen Terminalarten stehen. Erzeugen Sie /etc/termcap durch Abmagerung von /etc/termcap.src. Merken Sie sich den Kurznamen Ihres Terminals (z.B. pv für das von PCS gelieferte vt100), und tragen Sie in /etc/ttytype eine Zeile für dieses Terminal ein, z.B. "pv tty1".

Nach erneutem Einloggen muss die Shell-variable TERM gleich dem Kurznamen sein.

Terminal-Interface

Obwohl der QU68000 erheblich schneller als eine LSI ist, kann es passieren, dass manchmal bei hohen Baud-Raten am DLV11 Zeichen geschluckt werden. Das ist besonders bei den Funktionstasten während des Editierens festzustellen, bei denen oft drei Zeichen hintereinander mit voller Geschwindigkeit zum Rechner geschickt werden. Um den Effekt so klein wie möglich zu halten, wird empfohlen, die DLV11-Karten in der Daisy-chain ganz nach vorne zu stecken. Wenn zu häufig Zeichen verlorengehen, muss die Baudrate herabgesetzt werden. Terminals, bei denen die Cursortasten nur 1 Zeichen abschicken, sind besonders günstig.

Leider hat das DLV11 keine interne Zeichenpufferung, wohl aber das DZ11 und das DH11, die es auf dem Markt für 4, 8 und 16 Kanäle gibt. Ausserdem bieten diese Multiplexer eine Modemsteuerung. Dadurch wird beim Ausschalten des Terminals das Signal SIGHUP erzeugt, und beim Einschalten automatisch login gemacht. Nur mit DZ11 oder DH11 funktioniert der packet driver pk und damit uucp und uux. Diese Multiplexer sind aber leider ziemlich teuer.

Drucker

Im Kern ist ein Standard-Drucker-Interface generiert (LP11). Sie können manchmal auch einen seriellen Drucker an einen Terminalausgang anschliessen. Achten Sie dann darauf, dass dieses Terminal in /etc/ttys nicht erscheint, so dass für den Drucker keine Shell generiert wird. Mit stty können Sie Verzögerungen für den Wagenrücklauf einstellen, oder CR oder CR LF als newline einstellen. Siehe dazu I/4 tty.

Wir benutzen als Drucker einen Versatec V80. Dieser braucht leider sehr teures Spezialpapier und stinken tut er auch, aber er macht keinen Krach, bedruckt im Printmodus 1000 Zeilen in der Minute, und, was am besten ist, lässt sich als Ausgabegerät für den troff nutzen. Dieses Papier wurde auf dem V80 erstellt. Wir hoffen sobald wie möglich einen kleinen Laserdrucker, z.B. den Canon LBP10, anschliessen zu können.

Profile

In Ihrem login-directory sollte ein file mit dem Namen ".profile" existieren. Es enthält Kommandos, die die Shell beim login ausführt. Als Beispiel sehen Sie folgendes .profile-file:

```
HOME=/
PATH=:/usr/ucb:/bin:/usr/bin
MAIL=/usr/spool/mail/root
export PATH
```

Systemgenerierung

Wir kommen nun zur eigentlichen Systemgenerierung. Unser Ziel ist es, ein Programm /unix zu erzeugen, das ohne Dialogteil ist, und alle nötigen Treiber und nur diese enthält. Dazu müssen Sie im directory /usr/sys die Dateien c.c und l.s ändern, und zwar nur die Zeilen, in denen der Kommentar /* MOD */ erscheint. In c.c und l.s erreichen Sie durch Ein-oder Auskommentieren, dass der

betreffende Treiber mit dazugebunden wird oder nicht. Die anderen Änderungen in c.c verstehen sich hoffentlich von selbst. nldisp muss so gesetzt werden, dass in der Tabelle linesw der Multiplex-Treiber nicht mitgezählt wird. Wollen Sie auf den Multiplextreiber verzichten, dann löschen Sie in der library lib2 die Module mx1.o und mx2.o. Statt dieser wird dann das Modul fakemx.o zum Kern dazugebunden, dass nur dazu dient, die unbefriedigten Referenzen abzusättigen. Nach Andern der beiden Dateien sagen Sie "make". Nun werden die Dateien übersetzt und ein neues unix gebunden, das nach /nunix geschrieben wird. Booten sie /nunix, eventuell nachdem Sie es auf Floppy kopiert haben, und wenn es ordnungsgemäss läuft, benennen Sie es nach /unix um. Das Unix, das Sie normalerweise laden, muss den Namen /unix tragen, damit die Programme ps und ss funktionieren.

Wir gehen bei der Generierung anders vor als an der PDP. Dort wird ein unix mitgeliefert, das auf einer Minimalkonfiguration läuft. Wir fangen hingegen mit einem Maximal-unix (aunix) an, das, wie wir hoffen, auf allen Installationen zum Laufen gebracht werden kann. Es sollte trotzdem ein eigenes Unix erzeugt werden, um den lästigen Dialogteil loszuwerden, und um Speicherplatz für nicht benötigte Treiber freizugeben.

Special files

Wenn bei der Generierung ein Gerät nicht generiert wird, wird in den Tabellen in c.c ein leerer Eintrag (mit lauter nodev's) erzeugt, anstatt die betreffende Zeile ganz wegzulassen. Das hat den Vorteil, dass an allen Installationen die Treiber über dieselbe major device number angesprochen werden können. In Ihrem /dev-directory sollten alle special-file-entries stehen, und Sie sollten die nicht benötigten löschen. Siehe auch die Shell-Prozedur /mkdev. Nicht löschen dürfen Sie die Einträge für console, null, mem, kmem, tty. Der Eintrag für swap ist nur für die Programme ss und ps interessant. Der swap-Bereich wird von ps gefunden, indem es die Datei /dev/swap eröffnet, in der Symboltabelle von /unix nach der Adresse von swplo sucht, mit dieser Adresse in /dev/kmem liest, und mit dem gefundenen Wert von swplo als offset in /dev/swap liest.

Standalone-Programme

Im directory /sa stehen Programme, die auf der nackten Hardware laufen können. Dieselben Programme sollten auch auf Ihrer Boot-Floppy (ohne Prefix /sa) stehen. Diese Programme können entweder direkt vom Minitor geladen werden, oder vom Programm boot. Das Programm boot muss vom Minitor geladen werden, kann dann aber andere Programme von allen möglichen Geräten laden. Ausser boot existieren zur Zeit die Programme cat, mkfs und restor. Da die Programme ohne Parameter aufgerufen werden, muss ein Parameterdialog stattfinden. In diesem Dialog werden die Namen von Files einheitlich nach folgendem Muster gebildet:

dev(unit,offset)name

wobei dev gleich hb, rl, hk, rp, hp, rk, rx, tm oder ht ist, unit die physikalische (!) unit number angibt, offset ein physikalischer Offset in Blöcken (oder tapemarks bei tm und ht) ist, und name entweder leer oder ein normaler unix-Pfadname ist. Eine logische unit lässt sich als Kombination von physikalischer unit number und physikalischem Offset ansprechen. So ist z.B. die logische unit rl0 gleich rl(0,0), rl1 gleich rl(0,10240), rl2 gleich rl(1,0) und rl3 gleich rl(1,10240).
Beispiele:

hk(0,0)/sa/boot	hk0 (root device), offset 0, File /sa/boot
rl(1,10240)xxx	file xxx auf der zweiten Hälfte von RL-Platte 1
tm(0,3)	das dritte File auf dem Magnetband.
rx(3,0)	Floppy 1, double density

Probieren Sie, boot zu laden, mit Hilfe von boot cat zu laden, und mit cat das File /etc/rc auszugeben. An unserer Installation geht das wie folgt, von der boot-Floppy aus:

rx	< Minitor: Laden von Floppy
/boot	< Lade boot-Programm
g	< Start
boot :	> Prompt von boot
rx(2,0)cat	< Floppy 0, double density, Programm cat
File:	> Ausgabe von cat
hk(0,0)/etc/rc	< gewünschtes File
...	> folgt Ausgabe von /etc/rc
Exit called	> Ende von cat
boot :	> Neue Aufforderung von boot

Die Standalone-Programme werden im wesentlichen zum Erzeugen und Sichern von Filesystemen benutzt. Sie können mit mkfs eine neue oder defekte Platte formatieren und mit restor einen früheren Dump auf die Platte schreiben. Wenn Sie allerdings nur eine Festplatte haben, und keine Möglichkeit zur Datensicherung, werden Sie irgendwann grosse Probleme bekommen.

Datensicherung

Probleme mit der Datensicherheit entstehen in der Regel durch Unterschiede zwischen den Pufferblöcken im Speicher und den Blöcken auf der Platte. Wenn das System abgeschaltet wird, muss unbedingt vorher das Kommando sync gegeben werden, um alle geänderten Blöcke auf die Platte zurückzuschreiben. Im multiuser-Betrieb wird automatisch alle 30 Sekunden durch das Programm /etc/update ein sync gemacht.

Der Unix-Kern ist stabil genug, dass er bei uns den ganzen Tag durchläuft. Wenn das System doch einmal abstürzt, z.B. infolge memory parity, wird in der Regel noch ein sync durchgeführt. Wenn das System blockiert (sollte es natürlich nicht), betätigen Sie erst die Halt-Taste am Rechner, um so das System zu einem "ordnungsgemässen" Absturz zu bringen. Erst dann, oder wenn Sie sync gesagt haben, betätigen Sie die Init-Taste.

Beim Hochfahren in den Multi-user-mode und nach einem Absturz sollten Sie standardmässig fsck aufrufen, um Inkonsistenzen zu erkennen und zu beheben. Auch die Programme icheck, ncheck, dcheck und clri erfüllen noch ihren Zweck. Die Reparatur von Platten besprechen wir im nächsten Abschnitt.

Wenn eine Reparatur erfolglos ist, haben Sie hoffentlich immer schön Datensicherung gemacht. Am einfachsten ist es, eine Platte physikalisch auf eine andere zu kopieren. Haben Sie beispielsweise zwei mal RL02 Wechselplatten, können Sie die eine auf die andere (z.B. 0 auf 2, siehe oben) durch

```
dd if=/dev/rrl0 of=/dev/rrl2 bs=40b
```

kopieren. Durch das Kopieren in 40*512 byte langen Blöcken geht das sehr schnell.

Am elegantesten geht Datensicherung natürlich mit dem Magnetband. Benutzen Sie dazu die Programme tar, dump und cpio. Tar und cpio haben den Vorteil, dass Sie sehr leicht einzelne Files oder directories von dem Band runterholen können. Find mit option -cpio gibt ihnen die Möglichkeit, Files mit einer bestimmten Eigenschaft (z.B. jünger als ..., grösser als ...) auf Band zu schreiben. Dump bietet als Vorteil den inkrementellen dump und das standalone-restor. Der grosse Nachteil von tar ist, dass keine special-files mitgedumpt werden. Es sei darauf hingewiesen, dass man den Band-Programmen auch immer, wegen der Unix-Fileunabhängigkeit, statt eines Bandgeräts eine Platte angeben kann. Dumpen können Sie auch im laufenden Betrieb.

Sie müssen auf jeden Fall das root-filesystem mit dump sichern, um mit dem Standalone-Programm restor das System auch dann wiederherstellen zu können, wenn die Platte völlig zerstört ist. Die anderen Filesysteme lassen sich besser mit tar oder cpio sichern.

Um den Benutzern zu helfen, die Datensicherung nur auf Floppys betreiben können, haben wir das dump-Kommando um die Option **b** *size* erweitert. *size* ist dabei die Grösse der Floppy in Blöcken (500, 1001 oder 2002). Diese neue Option ist der Option **s** nachempfunden, die die Länge des Bandes in feet anzugeben erlaubt. Um also z.B. das filesystem rl0 auf double sided, double density Floppies zu dumpen, geben Sie das Kommando

```
dump 0ubf 2002 /dev/rrx2 /dev/rrl0
```

Das Programm dump wird automatisch eine neue Floppy anfordern, wenn die alte voll ist.

Reparieren von Filesystemen

Beim Plattenreparieren sollte die zu reparierende Unit nicht gemounted sein. Beim Root-device ist das nicht möglich. Hier arbeiten Sie am besten mit dem raw device, achten darauf, dass /etc/update nicht läuft, und betätigen nach Reparatur sofort die Init-Taste. Damit wird verhindert, dass fehlerhafte Blöcke aus dem Speicher die reparierte Platte wieder überschreiben. Beachten Sie beim Reparieren, dass das block device über den Unix-Puffer-Mechanismus angesprochen wird, während das raw device direkt auf der Platte arbeitet. Ein "sync" bewirkt, dass block und raw device konsistent werden, indem die geänderten Blöcke im Puffer auf die Platte zurückgeschrieben werden.

Die Reparatur mit Hilfe von fsck ist in der Dokumentation ausführlich beschrieben. Benutzen Sie clri, wenn ein Inode so zerstört ist, dass er unmögliche Blocknummern enthält. Icheck und ncheck können Sie benutzen, um herauszufinden, in welcher Datei ein bad block steht.

Herstellen einer Systemplatte

Wenn Sie ein neues System mit Software auf Floppys oder Band bekommen, oder wenn Sie Ihr root-filesystem zerstört haben, müssen Sie das root-filesystem neu aufbauen. Dazu sind folgende Schritte erforderlich:

1.

Formatieren Sie die Platte, falls erforderlich.

2.

Booten Sie mkfs, z.B. von Floppy, durch Eingabe an den Minitor: rx, /boot, g, rx(2,0)mkfs. Mkfs fragt nach Dateinamen und filesystem-Grösse.

3.

Wenn mkfs beendet ist ,laden sie restor: rx(2,0)restor. restor fragt nach dem Eingabefile. Legen Sie nun das dump-Band oder die erste dump-Floppy ein und antworten Sie: tm(0,0) bzw. rx(2,0). restor fragt noch einmal, ob Sie sicher sind: Last chance before scribbling on ... . Drücken Sie <Return>.

Einführende Lektüre

Aller Anfang ist schwer. Um Ihnen den Einstieg zu erleichtern, empfehlen wir, von Unix-Lehrbüchern abgesehen, das Studium der Manuale wie folgt: Zuerst die einführenden Artikel im Handbuch II: "Summary", "The Unix Time- Sharing System", "Unix for Beginners", "A tutorial Introduction to the Unix Text Editor", "An Introduction to the Unix Shell". Spielen Sie dann mit ein paar Kommandos herum, die in I/1 beschrieben sind, z.B. ls, echo, wc, who, bc, cat, date, find, grep etc. Lernen Sie anschliessend einen Bildschirmeditor, wobei wir den ped empfehlen, da er viel einfacher als vi ist. Für Leute, die tiefer ins System einsteigen wollen, geht es dann weiter mit II/29, II/31 und II/32, also "Setting up Unix", (Absätze über special files, disk layout, new users und multiple users), dann "Unix Implementation" und "The Unix I/O system". Diejenigen, die in C programmieren wollen, lesen dann II/14 und II/17, nämlich "The C Programming language" und, besonders wichtig, "Unix Programming". Wenn Sie soweit sind, werden die Kapitel 2 und 3 im Handbuch I interessant. Studieren Sie sorgfältig die header-files in /usr/include und /usr/include/sys. Vor allem, benutzen Sie sie auch, um die Konsistenz und Portabilität Ihrer Programme zu erhöhen. Bedenken Sie, dass unter Unix geschriebene Programme eine enorme Verbreitung erreichen können.

Wenn Sie dann eigene Programme schreiben, wird es Sie vielleicht interessieren, wie man diese debuggen kann: mit adb (II/18) und pbug. Läuft das Programm dann, muss es dokumentiert werden. Fangen Sie am einfachsten mit den ms-Macros an (II/8), und steigen Sie dann etwas tiefer in nroff/troff ein. Dabei lohnt es sich immer, die Original-Dokumentation in /usr/man und /usr/doc zu studieren. Werden Ihre Programme grösser, lernen und benutzen Sie make (II/16). Sind Sie bis dahin gekommen, lesen Sie alles nochmal durch und den Rest auch. Anschliessend würde ich Urlaub machen.

MUNIX V1.3 - Summary

PCS
Pfaelzer - Wald - Str. 36
8000 Muenchen 90

Software facilities available on MUNIX

Most of the programs available as MUNIX commands are listed in this summary. Object code and on-line manuals are distributed for all of the listed software.

Commands are self-contained and do not require extra setup information, unless specifically noted as ``interactive.'' Interactive programs can be made to run from a prepared script simply by redirecting input. Most programs intended for interactive use (e.g., the editor) allow for an escape to command level (the Shell). Most file processing commands can also go from standard input to standard output (``filters''). The piping facility of the Shell may be used to connect such filters directly to the input or output of other programs.

MUNIX is based on UNIX* V7. Extensions and changes to V7, which are visible to the user, are marked with an ``*''.

---

*UNIX is a Trademark of Bell Laboratories.

October 29, 1982

1. Basic Software

This includes the time-sharing operating system with utilities, a machine language assembler and a compiler for the programming language C. Enough software to write and run new applications.

1.1. Operating System

MUNIX — The basic resident code on which everything else depends. Supports the system calls, and maintains the file system. A general description of UNIX design philosophy and system facilities appeared in the Communications of the ACM, July, 1974. A more extensive survey is in the Bell System Technical Journal for July-August 1978. Capabilities include:

- Reentrant code for user processes.
- Separate instruction and data spaces.
- ``Group'' access permissions for cooperative projects, with overlapping memberships.
- Alarm-clock timeouts.
- Timer-interrupt sampling and interprocess monitoring for debugging and measurement.
- Multiplexed I/O for machine-to-machine communication.

*DRIVERS — All I/O is logically synchronous. I/O devices are simply files in the file system. Normally, invisible buffering makes all physical record structure and device characteristics transparent and exploits the hardware's ability to do overlapped I/O. Unbuffered physical record I/O is available for unusual applications. Drivers for the following devices and controllers are available:

- Asynchronous interfaces: DLV11, terminal multiplexer DZV11. Support for most common ASCII terminals.
- Parallel line printer: LPV11

- Serial line printer UD3 with DLV11j interface
- Versatec V80 printer plotter
- Magnetic tape: TU10
- Floppy disk: RX01/02
- Disks: RL01/02, RK05/06/07, RM02
- Honeywell Bull D120 with DRV 11 interface
- Physical memory of QU68000
- Null device

*DRIVER-MANIPULATION

Force drivers to swap bytes, for exchanging data with foreign systems, especially DEC systems.

- rxctrl: floppy driver
- tmctrl: tape driver

*FORMAT Procedures to format several devices.

BOOT Procedures to get MUNIX started.

*NEWCONF Make a new MUNIX kernel from an already existing or from an interactively created configuration file.

*WHATCONF Examine a MUNIX kernel, for what devices it is configured.

1.2. User Access Control

LOGIN Sign on as a new user.

- Verify password and establish user's individual and group (project) identity.
- Adapt to characteristics of terminal.
- Establish working directory.
- Announce presence of mail (from MAIL).
- Publish message of the day.

- Execute user-specified profile.
- Start command interpreter or other initial program.

PASSWD Change a password.

- User can change his own password.
- Passwords are kept encrypted for security.

NEWGRP Change working group (project). Protects against unauthorized changes to projects.

1.3. Terminal Handling

TABS Set tab stops appropriately for specified terminal type.

STTY Set up options for optimal control of the current output terminal. In so far as they are deducible from the input, these options are set automatically by LOGIN.

- Half vs. full duplex.
- Carriage return+line feed vs. newline.
- Interpretation of tabs.
- Parity.
- Mapping of upper case to lower.
- Raw vs. edited input.
- Delays for tabs, newlines and carriage returns.

*SETTY Set up options for optimal control of other terminals. Analogous to STTY.

1.4. File Manipulation

CAT	Concatenate one or more files onto standard output. Particularly used for unadorned printing, for inserting data into a pipeline, and for buffering output that comes in dribs and drabs. Works on any file regardless of contents.
*PACK	Pack many files to one or unpack them.
CP	Copy one file to another, or a set of files to a directory. Works on any file regardless of contents.
*PR	Print files with title, date, and page number on every page.
	# Multicolumn output.
	# Parallel column merge of several files.
TAIL	Print last n lines of input
	# May print last n characters, or from n lines or characters to end.
*LINENO	Print files with leading line numbers.
LPR	Off-line print. Spools arbitrary files to the line printer.
*VPR	Raster printer/plotter spooler.
*UPR	Spooler for the UD3 line printer.
*PG	Simple interactive display function for text files.
CMP	Compare two files and report if different.
SUM	Sum the words of a file.
SPLIT	Split a large file into more manageable pieces. Occasionally necessary for editing (ED).
*INTO	Copy standard input until end of file to standard output.
DD	Physical file format translator, for exchanging data with foreign systems, especially IBM 370's.

1.5. Manipulation of Directories and File Names

RM — Remove a file. Only the name goes away if any other names are linked to the file.

- Step through a directory deleting files interactively.
- Delete entire directory hierarchies.

LN — Link another name (alias) to an existing file.

MV — Move a file or files. Used for renaming files.

CHMOD — Change permissions on one or more files. Executable by files' owner.

CHOWN — Change owner of one or more files.

CHGRP — Change group (project) to which a file belongs.

MKDIR — Make a new directory.

RMDIR — Remove a directory.

CD — Change working directory.

FIND — Prowl the directory hierarchy finding every file that meets specified criteria.

- Criteria include:
  name matches a given pattern,
  Creation date in given range,
  date of last use in given range,
  given permissions,
  given owner,
  given special file characteristics,
  boolean combinations of above.
- Any directory may be considered to be the root.
- Perform specified command on each file found.

1.6. Running of Programs

SH — The Shell, or command language interpreter.

- Supply arguments to and run any executable program.
- Redirect standard input, standard output, and standard error files.
- Pipes: simultaneous execution with output of one process connected to the input of another.
- Compose compound commands using:
  if ... then ... else,
  case switches,
  while loops,
  for loops over lists,
  break, continue and exit,
  parentheses for grouping.
- Initiate background processes.
- Perform Shell programs, i.e., command scripts with substitutable arguments.
- Construct argument lists from all file names satisfying specified patterns.
- Take special action on traps and interrupts.
- User-settable search path for finding commands.
- Executes user-settable profile upon login.
- Optionally announces presence of mail as it arrives.
- Provides variables and parameters with default setting.

TEST — Tests for use in Shell conditionals.

- String comparison.
- File nature and accessibility.
- Boolean combinations of the above.

EXPR — String computations for calculating command arguments.

- Integer arithmetic
- Pattern matching

WAIT — Wait for termination of asynchronously running processes.

READ — Read a line from terminal, for interactive Shell procedure.

ECHO — Print remainder of command line. Useful for diagnostics or prompts in Shell programs, or for inserting data into a pipeline.

SLEEP — Suspend execution for a specified time.

NOHUP — Run a command immune to hanging up the terminal.

NICE — Run a command in low (or high) priority.

KILL — Terminate named processes.

CRON — Schedule regular actions at specified times.

- Actions are arbitrary programs.
- Times are conjunctions of month, day of month, day of week, hour and minute. Ranges are specifiable for each.

AT — Schedule a one-shot action for an arbitrary time.

TEE — Pass data between processes and divert a copy into one or more files.

*STKSIZ — Change the stacksize of a program.

1.7. Status Inquiries

*LS — List the names of one, several, or all files in one or more directories.

- # Alphabetic or temporal sorting, up or down.
- # Optional information: size, owner, group, date last modified, date last accessed, permissions, i-node number.

FILE — Try to determine what kind of information is in a file by consulting the file system index and by reading the file itself.

DATE — Print today's date and time. Has considerable knowledge of calendric and horological peculiarities.

- # May set UNIX 's idea of date and time.

DF — Report amount of free space on file system devices.

DU — Print a summary of total space occupied by all files in a hierarchy.

*DEVNM — Give the device name where a specified subdirectory is resident.

QUOT — Print summary of file space usage by user id.

WHO — Tell who's on the system.

- # List of presently logged in users, ports and times on.
- # Optional history of all logins and logouts.

TTY — Print name of your terminal.

PS — Report on active processes.

- # List your own or everybody's processes.
- # Tell what commands are being executed.
- # Optional status information: state and scheduling info, priority, attached terminal, what it's waiting for, size.

*SS Report on system status.

PWD Print name of your working directory.

1.8. Backup and Maintenance

MOUNT Attach a device containing a file system to the tree of directories. Protects against nonsense arrangements.

UMOUNT Remove the file system contained on a device from the tree of directories. Protects against removing a busy device.

MKFS Make a new file system on a device.

MKNOD Make an i-node (file system entry) for a special file. Special files are physical devices, virtual devices, physical memory, etc.

TP

TAR Manage file archives on magnetic tape.

- Collect files into an archive.
- Update DECtape archive by date.
- Replace or delete DECtape files.
- Print table of contents.
- Retrieve from archive.

*DUMP Dump the file system stored on a specified device, selectively by date, or indiscriminately. A multi-volume dump (i.e. on floppies) is possible.

RESTOR Restore a dumped file system, or selectively retrieve parts thereof.

SU Temporarily become the super user with all the rights and privileges thereof. Requires a password.

*FSCK File system consistency check and interactive repair.

DCHECK

ICHECK

NCHECK Check consistency of file system.

- Print gross statistics: number of files, number of directories, number of special files, space used, space free.
- Report duplicate use of space.
- Retrieve lost space.
- Report inaccessible files.
- Check consistency of directories.
- List names of all files.

CLRI Peremptorily expunge a file and its space from a file system. Used to repair damaged file systems.

SYNC Force all outstanding I/O on the system to completion. Used to shut down gracefully.

1.9. Accounting

The timing information on which the reports are based can be manually cleared or shut off completely.

AC Publish cumulative connect time report.

- Connect time by user or by day.
- For all users or for selected users.

SA Publish Shell accounting report. Gives usage information on each command executed.

- Number of times used.
- Total system time, user time and elapsed time.
- Optional averages and percentages.
- Sorting on various fields.

1.10. Communication

MAIL — Mail a message to one or more users. Also used to read and dispose of incoming mail. The presence of mail is announced by LOGIN and optionally by SH.

- Each message can be disposed of individually.
- Messages can be saved in files or forwarded.

CALENDAR — Automatic reminder service for events of today and tomorrow.

WRITE — Establish direct terminal communication with another user.

WALL — Write to all users.

MESG — Inhibit receipt of messages from WRITE and WALL.

CU — Call up another time-sharing system.

- Transparent interface to remote machine.
- File transmission.
- Take remote input from local file or put remote output into local file.
- Remote system need not be UNIX.

UUCP — UNIX to UNIX copy.

- Automatic queuing until line becomes available and remote machine is up.
- Copy between two remote machines.
- Differences, mail, etc., between two machines.

1.11. Basic Program Development Tools

Some of these utilities are used as integral parts of the higher level languages described in section 2.

AR — Maintain archives and libraries. Combines several files into one for housekeeping efficiency.

- Create new archive.
- Update archive by date.
- Replace or delete files.
- Print table of contents.
- Retrieve from archive.

*AS — M68000-Macro-Assembler, implemented by CERN/SIEMENS.

- Most powerful instruction set of available assemblers.
- Creates object program consisting of
  code, possibly read-only
  initialized data
  uninitialized data.
- Relocatable object code is directly executable without further transformation.
- Object code normally includes a symbol table.
- Multiple source files.
- Conditional assembly.

LIBRARY — The basic run-time library. These routines are used freely by all software.

- Buffered character-by-character I/O.
- Formatted input and output conversion (SCANF and PRINTF) for standard input and output, files, in-memory conversion.
- Storage allocator.
- Time conversions.

- Number conversions.
- Password encryption.
- Quicksort.
- Random number generator.
- Mathematical function library, including trigonometric functions and inverses, exponential, logarithm, square root, bessel functions.

*ADB Interactive debugger.

- Postmortem dumping.
- Examination of arbitrary files, with no limit on size.
- Interactive breakpoint debugging with the debugger as a separate process.
- Symbolic reference to local and global variables.
- Stack trace for C programs.
- Output formats:
  1-, 2-, or 4-byte integers in octal, decimal, or hex
  single and double floating point
  character and string
  disassembled machine instructions
- Patching.
- Searching for integer, character, or floating patterns.
- Handles separated instruction and data space.

OD Dump any file. Output options include any combination of octal or decimal by words, octal by bytes, ASCII, opcodes, hexadecimal.

- Range of dumping is controllable.

*XD Hexadecimal file dump.

*LD — Link edit. Combine relocatable object files. Insert required routines from specified libraries.

- Resulting code may be sharable.
- Resulting code may have separate instruction and data spaces.

LORDER — Places object file names in proper order for loading, so that files depending on others come after them.

NM — Print the namelist (symbol table) of an object program. Provides control over the style and order of names that are printed.

*SIZE — Report the core requirements of one or more object files.

STRIP — Remove the relocation and symbol table information from an object file to save space.

TIME — Run a command and report timing information on it.

PROF — Construct a profile of time spent per routine from statistics gathered by time-sampling the execution of a program.

- Subroutine call frequency and average times for C programs.

*MAKE — Controls creation of large programs. Uses a control file specifying source file dependencies to make new version; uses time last changed to deduce minimum amount of work necessary.

- Knows about CC, YACC, LEX, etc.

1.12. UNIX Programmer's Manual

MANUAL — Machine-readable version of the UNIX Programmer's Manual.

- System overview.
- All commands.
- All system calls.
- All subroutines in C and assembler libraries.

- # All devices and other special files.
- # Formats of file system and kinds of files known to system software.
- # Boot and maintenance procedures.

MAN — Print specified manual section on your terminal.

1.13. Computer-Aided Instruction

LEARN — A program for interpreting CAI scripts, plus scripts for learning about UNIX by using it.

- # Scripts for basic files and commands, editor, advanced files and commands, EQN, MS macros, C programming language.

2. Languages

2.1. The C Language

*CC — Compile and/or link edit programs in the C language. The UNIX operating system, most of the subsystems and C itself are written in C. For a full description of C, read ``The C Programming Language'', Brian W. Kernighan and Dennis M. Ritchie, Prentice-Hall, 1978.

- General purpose language designed for structured programming.
- Data types include character, integer, float, double, pointers to all types, functions returning above types, arrays of all types, structures and unions of all types.
- Operations intended to give machine-independent control of full machine facility, including to-memory operations and pointer arithmetic.
- Macro preprocessor for parameterized code and inclusion of standard files.
- All procedures recursive, with parameters by value.
- Machine-independent pointer manipulation.
- Object code uses full addressing capability of the M68000.
- Runtime library gives access to all system facilities.
- Floating point arithmetic realized by software.
- Definable data types.
- Block structure

LINT — Verifier for C programs. Reports questionable or nonportable usage such as:

Mismatched data declarations and procedure interfaces.
Nonportable type conversions.
Unused variables, unreachable code, no-effect operations.
Mistyped pointers.
Obsolete syntax.

- Full cross-module checking of separately compiled programs.

CB — A beautifier for C programs. Does proper indentation and placement of braces.

2.2. Other Algorithmic Languages

DC — Interactive programmable desk calculator. Has named storage locations as well as conventional stack for holding integers or programs.

- Unlimited precision decimal arithmetic.
- Appropriate treatment of decimal fractions.
- Arbitrary input and output radices, in particular binary, octal, decimal and hexadecimal.
- Reverse Polish operators:
  + - * /
  remainder, power, square root,
  load, store, duplicate, clear,
  print, enter program text, execute.

BC — A C-like interactive interface to the desk calculator DC.

- All the capabilities of DC with a high-level syntax.
- Arrays and recursive functions.
- Immediate evaluation of expressions and evaluation of functions upon call.
- Arbitrary precision elementary functions: exp, sin, cos, atan.
- Go-to-less programming.

*FFPCALC — Interactive programmable fast floating point desk calculator.

- Fortran expression syntax
- Optional assignement statement
- Two variables X and Y

- Functions: sqrt, power, log, exp, sin, cos, tan, atan, sinh, cosh, tanh, abs, neg, conversion to integer, rounding.

2.3. Macroprocessing

M4 — A general purpose macroprocessor.

- Stream-oriented, recognizes macros anywhere in text.
- Syntax fits with functional syntax of most higher-level languages.
- Can evaluate integer arithmetic expressions.

2.4. Compiler-compilers

YACC — An LR(1)-based compiler writing system. During execution of resulting parsers, arbitrary C functions may be called to do code generation or semantic actions.

- BNF syntax specifications.
- Precedence relations.
- Accepts formally ambiguous grammars with non-BNF resolution rules.

LEX — Generator of lexical analyzers. Arbitrary C functions may be called upon isolation of each lexical token.

- Full regular expression, plus left and right context dependence.
- Resulting lexical analysers interface cleanly with YACC parsers.

3. Text Processing

3.1. Document Preparation

ED — Interactive context editor. Random access to all lines of a file.

- Find lines by number or pattern. Patterns may include: specified characters, don't care characters, choices among characters, repetitions of these constructs, beginning of line, end of line.
- Add, delete, change, copy, move or join lines.
- Permute or split contents of a line.
- Replace one or all instances of a pattern within a line.
- Combine or split files.
- Escape to Shell (command language) during editing.
- Do any of above operations on every pattern-selected line in a given range.
- Optional encryption for extra security.

PTX — Make a permuted (key word in context) index.

SPELL — Look for spelling errors by comparing each word in a document against a word list.

- 25,000-word list includes proper names.
- Handles common prefixes and suffixes.
- Collects words to help tailor local spelling lists.

LOOK — Search for words in dictionary that begin with specified prefix.

CRYPT — Encrypt and decrypt files for security.

3.2. Document Formatting

*VTROFF

TROFF

NROFF

Advanced typesetting. TROFF drives a Graphic Systems phototypesetter; NROFF drives ascii terminals of all types. VTROFF drives a raster printer/plotter. TROFF, VTROFF and NROFF style is similar to RUFF (not available on MUNIX), but they are capable of much more elaborate feats of formatting, when appropriately programmed. TROFF, VTROFF and NROFF accept the same input language.

- All RUFF capabilities available or definable.
- Completely definable page format keyed to dynamically planted ``interrupts'' at specified lines.
- Maintains several separately definable typesetting environments (e.g., one for body text, one for footnotes, and one for unusually elaborate headings).
- Arbitrary number of output pools can be combined at will.
- Macros with substitutable arguments, and macros invocable in mid-line.
- Computation and printing of numerical quantities.
- Conditional execution of macros.
- Tabular layout facility.
- Positions expressible in inches, centimeters, ems, points, machine units or arithmetic combinations thereof.
- Access to character-width computation for unusually difficult layout problems.
- Overstrikes, built-up brackets, horizontal and vertical line drawing.
- Dynamic relative or absolute positioning and size selection, globally or at the character level.

- Can exploit the characteristics of the terminal being used, for approximating special characters, reverse motions, proportional spacing, etc.

The Graphic Systems typesetter has a vocabulary of several 102-character fonts (4 simultaneously) in 15 sizes. TROFF provides terminal output for rough sampling of the product.

NROFF will produce multicolumn output on terminals capable of reverse line feed, or through the postprocessor COL.

High programming skill is required to exploit the formatting capabilities of TROFF and NROFF, although unskilled personnel can easily be trained to enter documents according to canned formats such as those provided by MS, below. TROFF and EQN are essentially identical to NROFF and NEQN so it is usually possible to define interchangeable formats to produce approximate proof copy on terminals before actual typesetting. The preprocessors MS and TBL are fully compatible with TROFF, NROFF and VTROFF.

MS — A standardized manuscript layout package for use with NROFF/TROFFor VTROFF. This document was formatted with MS.

- Page numbers and draft dates.
- Automatically numbered subheads.
- Footnotes.
- Single or double column.
- Paragraphing, display and indentation.
- Numbered equations.

EQN — A mathematical typesetting preprocessor for TROFF. Translates easily readable formulas, either inline or displayed, into detailed typesetting instructions.

- Automatic calculation of size changes for subscripts, sub-subscripts, etc.
- Full vocabulary of Greek letters and special symbols, such as 'gamma', 'GAMMA', 'integral'.
- Automatic calculation of large bracket sizes.
- Vertical ''piling'' of formulae for matrices, conditional alternatives, etc.

- Integrals, sums, etc., with arbitrarily complex limits.
- Diacriticals: dots, double dots, hats, bars, etc.
- Easily learned by nonprogrammers and mathematical typists.

NEQN — A version of EQN for NROFF; accepts the same input language. Prepares formulas for display on any terminal that NROFF knows about, for example, those based on Diablo printing mechanism.

- Same facilities as EQN within graphical capability of terminal.

TBL — A preprocessor for NROFF/TROFF that translates simple descriptions of table layouts and contents into detailed typesetting instructions.

- Computes column widths.
- Handles left- and right-justified columns, centered columns and decimal-point alignment.
- Places column titles.
- Table entries can be text, which is adjusted to fit.
- Can box all or parts of table.

GREEK — Fancy printing on Diablo-mechanism terminals like DAS1-300 and DAS1-450, and on Tektronix 4014.

- Gives half-line forward and reverse motions.
- Approximates Greek letters and other special characters by overstriking.

COL — Canonicalize files with reverse line feeds for one-pass printing.

DEROFF — Remove all TROFF commands from input.

CHECKEQ — Check document for possible errors in EQN usage.

4. Information Handling

SORT — Sort or merge ASCII files line-by-line. No limit on input size.

- Sort up or down.
- Sort lexicographically or on numeric key.
- Multiple keys located by delimiters or by character position.
- May sort upper case together with lower into dictionary order.
- Optionally suppress duplicate data.

TSORT — Topological sort - converts a partial order into a total order.

UNIQ — Collapse successive duplicate lines in a file into one line.

- Publish lines that were originally unique, duplicated, or both.
- May give redundancy count for each line.

TR — Do one-to-one character translation according to an arbitrary code.

- May coalesce selected repeated characters.
- May delete selected characters.

DIFF — Report line changes, additions and deletions necessary to bring two files into agreement.

- May produce an editor script to convert one file into another.
- A variant compares two new versions against one old one.

COMM — Identify common lines in two sorted files. Output in up to 3 columns shows lines present in first file only, present in both, and/or present in second only.

JOIN — Combine two files by joining records that have identical keys.

GREP — Print all lines in a file that satisfy a pattern.

- # May print all lines that fail to match.
- # May print count of hits.
- # May print first hit in each file.

LOOK — Binary search in sorted file for lines with specified prefix.

WC — Count the lines, ''words'' (blank-separated strings) and characters in a file.

SED — Stream-oriented version of ED. Can perform a sequence of editing operations on each line of an input stream of unbounded length.

- # Lines may be selected by address or range of addresses.
- # Control flow and conditional testing.
- # Multiple output streams.
- # Multi-line capability.

AWK — Pattern scanning and processing language. Searches input for patterns, and performs actions on each line of input that satisfies the pattern.

- # Patterns include regular expressions, arithmetic and lexicographic conditions, boolean combinations and ranges of these.
- # Data treated as string or numeric as appropriate.
- # Can break input into fields; fields are variables.
- # Variables and arrays (with non-numeric subscripts).
- # Full set of arithmetic operators and control flow.
- # Multiple output streams to files and pipes.
- # Output can be formatted as desired.
- # Multi-line capabilities.

5. Novelties, Games, and Things That Didn't Fit Any Where Else

BANNER	Print output in huge letters.
BCD	Converts ascii to card-image form.
CAL	Print a calendar of specified month and year.
*FISH	Children's card-guessing game.
HANGMAN	Or hang. Word-guessing game. Uses the dictionary supplied with SPELL.
QUIZ	Test your knowledge of Shakespeare, Presidents, capitals, etc.
*STARTREK	Strategy game. Destroy the klingons.
*TWINKLE	twinkle1 or twinkle2. Milky way on the screen.
UNITS	Convert amounts between different scales of measurement. Knows about hundreds of units. For example, how many km/sec is a parsec/megayear?
*WORM	Lead the worm to random points.
*WORMS	Several worms running around on screen.
WUMP	Hunt the wumpus, thrilling search in a dangerous cave.
*XCHG	Give the hexadecimal code of any desired ASCII-character.

Deviations of MUNIX V1.3/M from MUNIX V1.3

PCS
Pfaelzer - Wald - Str. 36
8000 Muenchen 90

MUNIX/M is created for small systems with only 10MB mass storage. Therefore three voluminous subjects of MUNIX are omitted:

- LEARN, the computer-aided instruction package
- On-line manuals
- Utilities for a Versatec printer

October 29, 1982

Unterschiede von Munix zu UNIX† V7

ABSTRACT

Dieser Bericht beschreibt die Abweichungen unserer Implementierung von Unix V7 auf dem QU68000 gegenüber der PDP11 Version von Unix V7.

1. Warum wurde geändert?

Es gab mehrere gute Gründe, um Anderungen durchzuführen. Zunächst einmal wollten wir den grossen Adressraum des 68000 ausnutzbar machen, d.h. die 64k - Begrenzungen der PDP mussten fallen. Aufgrund der ganz anderen Segmentierung der QU68000 - Memory Management Unit waren wieder andere Anderungen durchzuführen. Auf der PDP in Assembler geschriebene Module mussten nach C oder (seltener) 68000 Assembler umgeschrieben werden. Die unterschiedliche Hardware wird natürlich auch in so maschinennahen Programmteilen wie der Unterbrechungsbehandlung und den daraus entstehenden signals sichtbar.

2. Was wurde geändert?

Unterschiede zur PDP11 Version ergeben sich auf 3 Arten: Software, die an der PDP, aber (noch) nicht am QU68000 läuft, Software, die am QU68000, aber nicht an der PDP läuft, und Software, die am QU68000 anders als an der PDP ist. Wir behandeln hier nur die Software, die nicht oder die anders läuft. Unsere Erweiterungen werden anderswo beschrieben. Dabei bemühen wir uns, die Anderungen schnellstmöglich in /usr/doc und /usr/man nachzuziehen.

3. (Noch) fehlende Software

Die folgende Liste hält sich an die Reihenfolge der Beschreibung im Unix - Manual I. Es wird immer eine kurze Begründung gegeben, warum etwas nicht läuft.

---

†UNIX is a Trademark of Bell Laboratories.

3.1. Programme (I/1)

arcv	Kein Bedarf, da keine V6 auf dem QU68000
bas	Leider in Assembler, hoffen anderswo ein Basic zu bekommen
enroll	Enroll, xsend und xget brauchen ein multiple precision package. Ausserdem kein Bedarf für Geheimhaltung bei uns
f77	Fortran wird zur Zeit portiert
factor	Factor und primes sind in Assembler geschrieben
lookall	Für lookall, pubindex, refer und lookbib bisher kein Bedarf
mkconf	Konfigurierung am QU68000 anders als an der PDP
ratfor	Ratfor und struct ohne Fortran sinnlos. Ratfor wird wahrscheinlich durch besseren Nachfolger EFL von Berkeley ersetzt
roff	In Assembler geschrieben, veraltet
tc	Tc und tk speziell für Tektronix 4014/4015, diese nicht vorhanden
uucp	Uucp und uux laufen, benötigen aber unbedingt eine DZ11-Hardware

3.2. System calls (I/2)

Ausser phys laufen alle system calls, auch mpx und pkon. phys wird nirgendwo verwendet. Sollte ein ähnlicher system call notwendig werden, lässt er sich leicht einbauen.

3.3. Library routines (I/3)

Der packet driver simulator ist in Wirklichkeit nicht Bestandteil der standard library, sondern von uucp. Am QU68000 ist der packet driver Teil des Systems (siehe oben).

3.4. Gerätetreiber (I/4)

Prinzipiell müssten alle in Handbuch I Kapitel 4 genannten Treiber funktionieren, oder sehr schnell zum Funktionieren gebracht werden können. Darüber hinaus haben wir noch Treiber, die in Kap. 4 nicht aufgeführt sind, vor allem den RX02 Treiber. Getestet sind natürlich nur die Treiber, für die wir eine Hardware zur Verfügung hatten. An Platten bedienen wir zur Zeit RL01/02, RK05 und RK06/07. Die Umstellungszeit für den RK05-Treiber betrug z.B. nur wenige Minuten. Die Treiber müssen jeweils etwas geändert werden, um über die DMA-Extension-Register den ganzen Speicher per DMA adressieren zu können.

4. Anderungen

Im folgenden wird, wiederum in der Reihenfolge von Handbuch I, beschrieben, wo Unterschiede zu V7 sind. Geändert wurden im wesentlichen maschinennahe Programme und header-files.

4.1. Programme (I/1)

adb Standardmässig erfolgt die Ausgabe von Variablen und Parametern hexadezimal anstatt octal. Der Befehl $C (C backtrace) gibt von jeder Variablen die ersten 4 bytes hexadezimal aus. Bei Variablen mit einer Länge n kleiner als 4 bytes heisst dies, dass nur die ersten n bytes zu beachten sind, im Falle von Registervariablen jedoch die letzten n bytes. Registervariablen werden durch ein "<" vor dem Namen kenntlich gemacht. Breakpoints am Anfang einer Prozedur sollten den Offset 10 relativ zum Prozeduranfang haben, damit bei Erreichen des Breakpoints das neue Stackframe fertig ist. Bitte beachten Sie die Disassembler-Funktion i des Adb!

as Der Assembler wurde am Cern (Hr. von Eicken) in Pascal geschrieben. Sourcecode und Dokumentation sind Bestandteil dieser Distribution. Der Aufruf erfolgt einfach durch as *filename*

cc Die Compileroption **-O** wird ignoriert. Es gibt neue Optionen **-L** (Zeilennummern werden im Code mitgeführt) und **-a** *offset* (im Assemblerlisting beginnt pc bei *offset*). Bei Namen sind die ersten 15 Buchstaben signifikant. Die Option **-4** bestimmt, dass integers die Länge 4 bytes haben. Default ist sizeof(int) == 2. Die Sourcen von Compiler und Binder sind Bestandteil dieser Distribution.

ld Die Binderoptionen **-n** und **-i** werden ignoriert. Das erzeugte Programm ist immer "pure" und "shareable". Neu hinzugekommene Optionen unterscheiden zwischen dem Binden von Benutzerprogrammen, die unter Unix ablaufen, und Programmen, die auf der nackten Maschine ablaufen sollen. Ferner kann eine Liste der Symbole erstellt werden.

make Make "kennt" Pascal-files, d.h. es ruft automatisch den Pascal-Compiler auf, um eine Datei mit Suffix ".p" zu übersetzen.

man Wir verwenden jetzt das man von Berkeley. Dabei wird die Dokumentation vorformatiert in directories parallel zu den /usr/man-directories gehalten. Auf diese Weise wird die Manual-Page sehr schnell ausgegeben.

pr pr hat eine neue Option **-n** , die ein File mit Zeilennummern ausgibt.

size Das size-Kommando gibt als vierte size die Stacksize des Programms aus. Der Wert 0 steht für die default-stacksize von 2 kbyte.

4.2. System calls (I/2)

brk Ein neuer systemcall lsbrk wird wie sbrk benutzt, hat aber einen long-Parameter

exec Achten Sie bitte darauf, dass eine 0 als Parameter als (char *)0 oder NULL geschrieben wird.

profil Die Nummer des Wortes in buff, dass erhöht werden soll, berechnet sich als (pc - offset) >> scale anstatt (pc - offset) * "0.scale".

ptrace Der Parameter data ist long. Im Falle request=1 bis 5 werden aber nur shorts gelesen oder geschrieben. Bei request=6 wird entsprechend dem layout von u.u_exvec mit Länge 4 geschrieben. Ein neuer Request 10 schreibt an die Adresse addr den ganzen u.u_exvec. Siehe in /usr/include/sys die files user.h und reg.h.

signal Es sind neue signals hinzugekommen. Siehe signal.h in /usr/include.

wait siehe exec

4.3. Library routines (1/3)

abort abort generiert signal SIGILL (illegal instruction).

putw putw und getw schreiben die bytes in der 68000-Reihenfolge, d.h. das MSB zuerst. Zusätzlich gibt es Routinen getl und putl für longs.

4.4. Treiber

Wichtig ist nur, dass beim Terminal die Steuerzeichen anders sind. Interrupt ist Ctrl-C statt DEL, quit ist Ctrl-Y statt FS, eofc ist Ctrl-Z anstatt Ctrl-D, ein Zeichen zu löschen geht mit DEL oder BS anstatt #, die ganze Zeile zu löschen geht mit Ctrl-X anstatt @.

4.5. File-Formate und header-files

a.out In a.out sind die unsigned Felder jetzt long. An magic numbers wird nur 0407 für Module und 0411 für fertige Programme erzeugt.

5. Floating Point

Zur Zeit wird Floating Point vom Prozessor nicht unterstützt. Erst der 68020 wird einen Floating Point Coprozessor haben. Bis dahin bietet Motorola ein "Fast Floating Point Package" an. Dieses kennt nur 32-bit floating point mit 24 bit Mantisse und 7 bit Exponent. Der C-Compiler behandelt deshalb den typ double zur Zeit genauso wie den typ float. Um das FFP-Paket dazuzubinden, geben Sie beim cc-Kommando die Option -f an.

6. Änderungen am Kern

6.1. Beschränkung der Programm-Grösse

Der QU68000 zerlegt den Adressraum in 16 Segmente à 1 Megabyte. Für den Benutzer sind die Segmente 6 bis 14 reserviert. Zur Zeit wird jeweils ein Segment für Code, Daten und Stack angelegt, d.h. von der MMU (Memory Management Unit) her kann ein Benutzer-Programm drei mal 1 Mbyte gross sein. Bei Bedarf können wir den Kern leicht so verändern, dass z.B. der Datenbereich bis zu 7 Segmente belegen kann, auf Kosten eines langsameren context-switchs. Tatsächlich gibt es aber zur Zeit eine Beschränkung, die verhindert, dass die Summe von Stack und Daten grösser als etwa 250 kbyte werden kann. Diese Obergrenze entsteht wie folgt:

Ein DMA-Gerät mit einem 18-bit Controller kann mit Hilfe der DMA-Extension-Register am QU68000 zwar den gesamten Speicher adressieren, kann aber keinen Transfer über eine 256kb-Grenze hinweg durchführen. Um nun zu verhindern, dass ein Segment mit Code oder Daten eine solche Grenze überlappt, wird zur Zeit ganz einfach bei der Erstellung der Freispeicherliste alle 256 kb ein kleiner Bereich von 512 bytes freigelassen. Damit kann kein zusammenhängender Speicherplatz von mehr als 255 ½ kb reserviert werden. Diese Methode werden wir dann durch etwas besseres ersetzen, wenn das Limit von 256kb irgendwo schmerzhaft spürbar wird.

6.2. Stackoverflow

Der 68000 hat bisher noch keine bus-error-recovery. Damit ist ein automatisches Erweitern des Stacks bei Stackoverflow sehr schwer. Stattdessen stellen wir das Kommando stksiz zur Verfügung (Aufruf stksiz *size file* ...), das das Programm *file* mit einem Stack der Länge *size* lädt. Defaultmässig hat der Stack die Länge 2 kb. Stürzt das Programm wegen Stackoverflow ab, müssen Sie stksiz mit einer grösseren *size* erneut aufrufen. Stackoverflow erkennen Sie mit dem adb, wenn bei $r (show registers) der Wert von ns (exception number + cpu state) in den ersten zwei bytes gleich 2 ist (exception 2: bus error), der Wert von ac (access address) etwas kleiner als 0xEFFFFF ist, und bei $m (show maps) der Wert von b2 in der "/ map (core)" etwas grösser als ac ist.

Wenn der 68010 da ist, wird der Stack automatisch verlängert werden.

6.3. Datenaustausch

Um den Datenaustausch mit anderen Unix-Installationen zu vereinfachen, kann beim Lesen und Schreiben von Floppy und Band die Reihenfolge der Bytes vertauscht werden. Die Programme tmctrl für Band und rxctrl für Floppy haben einen Parameter -s, der das Vertauschen der bytes einschaltet. Der Parameter -i bei rxctrl schaltet auf der Floppy das Interleaving der Sektoren aus. Der Parameter -f formatiert die Floppy. Der Parameter -d stellt bei beiden Programmen die defaults wieder her. Zum Datenaustausch auf Band sollte das Programm tar benutzt werden.

Das Programm "mt" zum Manipulieren von Bändern hat zusätzliche Parameter sw und nsw (swap und noswap), die byteswapping ein- und ausschalten. Das Programm cpio hat einen (nichtdokumentierten) Parameter s, der gleichfalls bytes swappt. Cpio-Baender von PDP oder VAX können mit " mt nsw; cpio -iBsvd filename </dev/rmt0 " gelesen werden.

7. Blick in die Zukunft

Wir gehen davon aus, dass die Entwicklung von Hardware und Software für den QU68000 in nächster Zeit sehr stürmisch verlaufen wird. An Hardware- Entwicklung haben wir folgende Meilensteine zu erwarten:

Ein schneller lokaler Speicher wird die Prozessorgeschwindigkeit verdoppeln. Der neue Speicher wird auf die Software keinerlei Auswirkungen haben. Wenn die Hardware-floating-point-unit verfügbar ist, werden wir sehr schnell die restlichen Unix-Programme zum Laufen bringen. Insbesondere hoffen wir, Fortran bald zum Laufen zu bringen. Eine zweistufige Memory Management Unit mit paging wird ein echtes virtuelles Unix möglich machen. Wenn der 68010 verfügbar ist, wird ein wachsender Stack automatisch verlängert werden. Bitmap-Displays und lokale Netze werden die Benutzer-Umgebung von Unix fundamental ändern.

Die Weiterentwicklung der Software wird sich hoffentlich so abspielen, dass sich in steigendem Masse die Universitäten daran beteiligen. In naher Zukunft zu erwartende Projekte sind Tex/Metafont, Lisp, Prolog, Modula 2, Basic, Cobol, Fortran, paging unix, Adaptionen von System III und Berkeley Unix, Netzwerk-Software, Bitmap-Software, GKS, ADA, Emacs, Ingres etc.

Assembler 68000 User's Guide
Version 1.1
July 1982

ABSTRACT

This user's guide is intended for use in writing and translating assembly programs on 68000 UNIX† systems. The assembler is upward compatible with the M68000 Cross Macro assembler provided by Motorola. This guide restricts itself therefore to describing the extensions provided by us.

The Assembler 68000 is a modified version of the CERN Cross Macro Assembler M68MIL implemented in Pascal. Changes were made to write a.out format (the object module format coming with UNIX) instead of CUFOM (CERN Universal Format for Object Modules).

†UNIX is a Trademark of Bell Laboratories.

CONTENTS

1. INTRODUCTION

The Cross Macro Assembler described in this manual is derived from the CERN Cross Macro Assembler M68MIL but writes a.out object format instead of CUFOM. Some of the pseudo instructions were changed to provide the user with a means to control a.out segments instead of CUFOM sections. All other pseudo instructions, all machine instructions and the expression rules - as far as symbol types are not concerned - are the same as for M68MIL and Motorola's Cross Assembler.

The assembler translates assembler source programs for the Motorola 68000 microprocessor into a.out, the format of UNIX loadable modules. A linkage editor subsequently allows the combination and linking of several such modules into a new a.out module. An archiver (see *ar(1)*) permits the construction of a.out libraries, which can be placed in the input to the linkage editor. Besides, the assembler will accept source files 'piped' through the UNIX preprocessor *cpp*.

The assembler is upward compatible with the M68000 Cross Macro Assembler provided by Motorola. Additional pseudo instructions are provided to allow the generation of relocatable object modules. The user is advised to see the following Motorola publications:

M68000 Cross Macro Assembler Reference Manual
B68KXASM(D3), Third Edition, September 1979

MC68000 16-Bit Microprocessor, User's Manual
MC68000UM(AD2), Second Edition, January, 1980

and the UNIX documentation

UNIX Programmer's Manual, Volume 1 (especially *ar(1)*, *ld(1)*, *a.out(5)*)

as a base for the use of this assembler. This manual will restrict itself to the description of the extensions made to the definitions of the Motorola cross assembler. For the readers' convenience this will be done in complete chapters rather than by listing the explicit differences.

The main areas covered by this note are:

Short description of a.out
Expressions (generalized)
Pseudo instructions
Macro definitions
How to use the assembler

Acknowledgement: I would like to thank Mr. Horst von Eicken who gave us the Pascal sources and user manual of his CERN Cross Assembler. If you are interested in this assembler, please contact him at:

Horst von Eicken
Data Handling Division
CERN
CH 1211 Geneva 23
Switzerland

2. SHORT DESCRIPTION OF a.out

In a.out modules code and data fall into three segments: the text segment, the data segment and the bss segment. The **text** segment is the one in which the assembler begins, and it is the one into which instructions are typically placed. The UNIX system can enforce the purity of the text segment of programs by trapping write operations into it (see *ld(1)*).

The **data** segment is available for placing data or instructions which will be modified during execution. Anything which may go in the text segment may be put into the data segment. In programs with write-protected, shareable text segments, data segment contains the initialized but variable parts of a program.

The **bss** segment may not contain any explicitely initialized code or data. The length of the bss segment (like that of text or data) is determined by the high-water mark of the location counter within it. At the start of execution of a program, the bss segment is set up by statements such as:

```
lab  ds.l  2
```

Another a.out convention concerns the entry point to a program: a program starts at a label __entry (which is typically defined by some runtime system and does some basic initialization) and branches then to a user-defined label _main. The user has to make sure that one and only one _main label is defined within her/his program.

☛ Since a.out modules are always relocatable it is not possible in an a.out module to allocate a label (symbol) at a certain absolute address - as it can be reached elsewhere by means of an **org** pseudo instruction - rather than by arranging the link-edit input appropriately (see *ld(1)*). **org** is supported for compatibility but it's limited.

3. SYMBOLS AND EXPRESSIONS

3.1. Symbols

Symbols recognized by the assembler consist of one or more characters, the first sixteen of which are significant. The first character must be a letter (a through z and A through Z) or an underscore (_), each remaining character may be a letter, a digit (0 through 9) or an underscore. The names for registers (**a0** through **a7**, **d0** through **d7**,**sp**,**usp**,**ccr**,**sr**), instructions (**abcd .. unlnk**) and pseudo instructions (**blong .. ttl**) are predefined symbols and may not be redefined by the user.

☛ **a** and **A** are different; predefined names must be written in lower case.

Numbers recognized by the assembler include decimal, hexadecimal and octal values. Decimal numbers are specified by a string of decimal digits (0 through 9); hexadecimal numbers are specified by a dollar sign ($), followed by a string of hexadecimal digits (0 through 9, A through F, a through f); octal numbers are specified by a colon (:), followed by a string of octal digits (0 through 7).

One or more characters enclosed by apostrophes (') constitute a character string. Character strings are left-adjusted and zero-filled (if necessary), whether stored or used as immediate operands. Only strings of four or fewer characters may be used as immediate operands. (In order to specify an apostrophe within a character string, two successive apostrophes must appear where the single apostrophe is intended to appear.)

The assembler has six types of symbols:

absolute symbol:

1. The symbol is equated (**equ**) or **set** to an absolute value.
2. The symbol is equated (**equ**) or **set** to a constant. Its value is unaffected by any possible future applications of the link- editor to the module.

text symbol:

1. The symbol is equated (**equ**) or **set** to a text symbol.
2. The symbol is defined in the text segment of the program. Its value is measured with respect to the beginning of the text segment of the program. If the assembler output is link-edited, its text symbols may change in value since the module need not be the first in the link editor's output. At the start of an assembly the value of "*" is text 0.

data symbol:

1. The symbol is equated (**equ**) or **set** to a data symbol.
2. The symbol is defined in the data segment of the program. Its value is measured with respect to the beginning of the data segment of the program. If the assembler output is link-edited, its data symbols may change in value since the module need not be the first in the link editor's output. After the first **data** pseudo instruction the value of "*" is data 0.

bss symbol:

1. The symbol is equated (**equ**) or **set** to a bss symbol.
2. The symbol is defined in the bss segment of the program. Its value is measured with respect to the beginning of the bss segment of the program. Like text and data symbols, the value of a bss symbol may change during a subsequent link-editor run, since previously loaded programs may have bss segments. After the first **bss** pseudo instruction the value of "*" is bss 0.

external symbol:

The symbol is listed in a **extern** pseudo instruction and is not defined in the current assembly. Its value is set to zero and must be defined during a subsequent link-editor run.

undefined symbol:

The symbol is neither defined in the current assembly nor listed in an **extern** pseudo instruction. The occurrence of such a symbol is indicated as an error.

All symbols except absolute symbols are relative, i.e. they represent relocatable addresses. Whenever the assembler encounters a text, data, or bss symbol it will generate a direct address (see below) and related relocation information. To yield a program counter relative address write

<rsym>(**pc**) or
<rsym>(**pc**,<reg>)

respectively.

Symbols defined within an assembly as absolute, text, data, or bss symbol may be exported by occurring in a entry pseudo instruction, i.e. their value and their type are available to the link-editor so that the program may be loaded with others that reference these symbols.

3.2. Expressions

An expression is a combination of symbols, constants, algebraic operators, and parentheses. The expression is used to specify a value which is to be used as an operand. Expressions follow the conventional rules of algebra.

Expressions may contain relative symbols. However the following rules must be followed in order for the expression to be valid:

1. Relative symbols or expressions cannot be multiplied, divided, added, or operated on with the logical operators.
2. A relative symbol or expression may have an absolute value added to or subtracted from it. The result is relative.
3. A relative symbol or expression may be subtracted from another relative symbol or expression provided they are both defined and of same type. The result is absolute.

3.3. Direct Addressing

(The addressing mode discussed here is called 'absolute addressing mode' in the related Motorola documentation. Since those address values cannot be changed at run time, but may be changed during a link-edit run, they are referred to here as direct addresses to avoid confusion with absolute symbols

defined ealier in this manual.)
Since the M:68000 microprocessor allows two forms of direct addressing,

- short direct address (16 bit address) -
- long direct address (32 bit address) -

the assembler has to take a decision as to which form to assume whenever it encounters a forward referenced absolute symbol, i.e. a symbol which has not yet been defined, or a relative symbol, i.e. a symbol the value of which may be changed by the link-editor. By default it will use the long direct address to ensure correct handling of the symbol. The pseudo instructions **fshort, flong, text, data** and **bss** will allow the programmer to override the assembler defaults. Assembling external symbols it will always use the long direct address mode.

A similar problem exists for branches. If a forward reference is found in a branch instruction, the assembler will use the two-word form of the instruction. Using the suffix .s for the branch instruction the programmer can force the assembler to generate the one-word form of the branch instruction. The pseudo instructions **bshort** and **blong** allow additional control.

4. PSEUDO INSTRUCTIONS

Pseudo Instructions discussed in this chapter are classified according to application as follows:

Module identification **(ident, end)**
Segment control **(text, data, bss)**
Symbol definition **(equ, set)**
Module linkage **(entry, extern)**
Data generation and storage reservation **(dc, ds, org)**
Conditional assembly **(else, endc, endif, ifeq, ifne)**
Source stream control **(insert)**
Listing control **(list,g,nolist,page,nopage,sttl,ttl,fail)**
Object code control **(blong, bshort, flong, fshort, noobj)**
Cross reference control **(xrefon, xrefoff)**

The next chapter describes the definition and use of macros. The format description for the pseudo instructions uses symbols which have the following syntactical meaning:

{..} Enclose optional fields of the pseudo instruction.

<..> Enclose a 'syntactic variable', e.g. <number>.

4.1. Module Identification

4.1.1. IDENT - Module Identification

For compatibility with M68MIL an **ident** pseudo instruction is accepted but is ignored. It has the following format:

ident <symbol>

<symbol>
The symbol defines a name (originally that of the module).

4.1.2. END - End of Module

An **end** pseudo instruction must be the last instruction of each module. It causes the assembler to terminate all counters, conditional assembly, or macro generation. It also causes the a.out module to be terminated.

end {<symbol>}

<symbol>
An optional symbol is accepted - for compatibility - but ignored. The program main entrypoint is determined by the label _main as stated in the a.out description.

4.2. Segment Control

Segment control pseudo instructions allow the programmer to divide a source module into separately controlled regions of a program, providing her/him with a means of changing type and value of the location counter. They start or resume assembly for one of the three a.out segments. The segment in use is the segment into which code is subsequently assembled. By default the assembler will always start with the text segment using long direct address mode (see 3.3). If the suffix **.s** is appended to a segment control pseudo instruction, the assembler will assume that this segment will finally be loaded into low address

memory and use direct short addresses for backward references.

☛ The suffix **.s** does not imply that forward references in that segment will also be resolved with direct short addresses. **fshort** must be used for this purpose.

4.2.1. Text Segment

text{.s}

Description:
The pseudo instruction causes the assembler to start or resume assembly in the text segment.

4.2.2. Data Segment

data{.s}

Description:
The pseudo instruction causes the assembler to start or resume assembly in the data segment.

4.2.3. Bss Segment

bss{.s}

Description:
The pseudo instruction causes the assembler to start or resume storage allocation in the bss segment (uninitialized data: **ds** and **org** only).

4.3. Symbol Definition

4.3.1. EQU - Equate Symbol Value.

<symbol> **equ** <expression>

<symbol>
A location symbol following the naming rules must be defined.

<expression>
An expression following the expression rules. Forward references are not allowed.

Description:
An **equ** pseudo instruction permanently defines the symbol in the location field as having the value and type indicated by the expression in the variable field.

4.3.2. SET - Set or reset symbol value.

<symbol> **set** <expression>

<symbol>
A location symbol following the naming rules must be defined.

<expression>
An expression following the expression rules. Forward references are not allowed.

Description:
A **set** pseudo instruction defines the symbol in the location field as having the value and type indicated by the expression in the variable field. A subsequent **set** using the same symbol redefines the symbol to the new value and type.

4.4. Module Linkage

The pseudo instructions **entry** and **extern** do not define symbols but either declare symbols defined within a module as being available outside the module or declare symbols referred to in the module as being defined outside the module.

4.4.1. ENTRY - Declare Entry Symbols

entry <sym_1>, <sym_2>,..., <sym_n>

<sym_i>
Linkage symbol. Each symbol must be defined in the module as nonexternal (must not be listed on an **extern** pseudo instruction).

Description:
The **entry** pseudo instruction specifies which of the symbolic addresses defined in the module can be referred to by modules assembled independently; **entry** lists entry points to the current module.

4.4.2. EXTERN - Declare External Symbols

extern <sym_1>, <sym_2>,..., <sym_n>

<sym_i>
Linkage symbol. These symbols must not be defined within the module.

Description:
The **extern** pseudo instruction lists symbols that are defined as entry points in independently assembled modules for which references can appear in the module being assembled.

4.5. Data Generation and Storage Reservation

4.5.1. DC - Define Constant

{<symbol>} **dc** <opr_1>, <opr_2>,..., <opr_n>
{<symbol>} **dc.b** <opr_1>, <opr_2>,..., <opr_n>
{<symbol>} **dc.w** <opr_1>, <opr_2>,..., <opr_n>
{<symbol>} **dc.l** <opr_1>, <opr_2>,..., <opr_n>

<symbol>
A symbol following the naming rules may be defined.

<opr_i>
The operand can be a symbol, an ascii, decimal or hexadecimal value or an expression evaluating to such a value.

Description:
The function of the **dc** pseudo instruction is to define a constant in memory. The **dc** directive may have one operand, or multiple operands which are separated by commas. The operand field may contain a value (decimal, octal, hexadecimal, or character string), a symbol or a expression. The constant is aligned on a word boundary if word(.w) or long (.l) is specified, or a byte boundary if byte (.b) is specified. The constant is limited to 60 bytes.

The following rules apply to size specifications on character strings:

dc.b If an odd number of bytes (characters) are entered, the odd byte on the right will be zero filled unless the next source statement is another **dc.b** or **ds.b**. In this case the next **dc.b** or **ds.b** will start in the odd byte on the right.

dc.w If an odd number of bytes (characters) are entered, the last word will be zero filled on the right to force an even byte count.

dc.l If less than a multiple of four bytes are entered, the last long word will be zero filled on the right to a multiple of four bytes.

4.5.2. DS - Define Storage

{<symbol>} **ds** <expression>
{<symbol>} **ds.b** <expression>
{<symbol>} **ds.w** <expression>
{<symbol>} **ds.l** <expression>

<symbol>
A symbol following the naming rules may be defined.

<expression>
The expression must evaluate to an absolute, positive value.

Description:
The **ds** pseudo instruction is used to reserve memory locations. The contents of the memory reserved are zero filled for text and data segment, for bss segment they are not initialized in any way. The expression must evaluate to an absolute value. Forward references are not allowed.

4.5.3. ORG - Origin

Since a.out modules are always relocatable the **org** pseudo instruction is applicable only in a very restricted manner: it can be used to advance the location counter a certain number of bytes or to a certain label. The skipped locations are zero filled.

org <expression>

Description:

The **org** pseudo instruction is used to advance the location counter <expression> bytes. It's identical with

ds.b <expression>*

i.e. the expression must be of current segment type and evaluate to an value greater than the actual location counter.

4.6. Conditional Assembly

The pseudo instructions **ifeq** and **ifne** permit optional assembly or skipping of source code. The instructions immediately following the test instruction are assembled if the tested condition is true and skipped if the condition is false. Skipping is terminated either by a source statement count on the **if** instruction, or by an **endif, else** or an **end**. The statement count, when used, is decremented for instruction lines only; comment lines (identified by * in column one) are not counted.

The result of an **if** test is determined by the value of the expression in pass one of the assembler; the value of a relative symbol is relative to the origin of the segment in which it was defined. The value of an external symbol is zero if the symbol was declared as external. If the symbol was defined relative to a declared external, the value is the relative value. **if**'s may be nested up to ten levels deep.

4.6.1. ENDIF - End of IF Range

{<if_name>} **endif**

<if_name>

An optional symbol; defines the name of an **ifeq, ifne** or **else** sequence; or blank.

Description:

An **endif** pseudo instruction (or **endc** for compatibility with the Motorola assembler) causes termination of skipping and assembly to resume. When the sequence containing the **endif** is being assembled, or is controlled by a statement count, the **endif** has no effect other than to be included in the count.

Skipped instructions such as macro references are not expanded. Thus, any **endif** that would have resulted from an expansion is not detected.

Skipping of a sequence initiated by an **ifeq, ifne** or **else** that is assigned a name is terminated by an **endif** specifying the same name. Skipping of a sequence initiated by an unnamed **ifeq, ifne** or **else** is terminated by an unnamed **endif**.

4.6.2. ELSE - Reverse Effects of IF.

{<if_name>} **else**

<if_name>
An optional symbol; defines the name of an **ifeq, ifne** or **else** sequence; or blank.

Description:
By means of the **else** instruction, the assembler provides the facility to reverse the effects of an **if** test within the **if** range. An **else** detected during skipping causes assembly to resume at the instruction following the **else**. An **else** detected while a sequence is being assembled initiates skipping of source code following the **else**. Skipping continues until either an **end** or an **endif** for the sequence is detected.

An **else** specifying the sequence by name terminates skipping of a sequence initiated by an **ifeq** or **ifne** with the same name. An unnamed **else** terminates skipping of a sequence initiated by an unnamed **ifeq** or **ifne**. Skipped instructions such as macro references are not expanded; any **else** that would have resulted from the expansion is not detected.

4.6.3. IFEQ - Test Expression is Equal Zero.

4.6.4. IFNE - Test Expression is Not Equal Zero.

{<if_name>} **ifeq** <expression>{,<line_count>}
{<if_name>} **ifne** <expression>{,<line_count>}

<if_name>
An optional symbol, defines the name of the **ifeq** or **ifne** sequence; or blank.

<expression>
A simple expression without forward reference. If the expression is erroneous, an error message is printed and assembly continues with the next instruction.

<line_count>
Optional absolute value specifying an integer count of the number of statements to be skipped.

Description:
The **ifeq** and **ifne** pseudo instructions test the value of the expression and assemble instruction in the **if** range when the condition is satisfied.

The <line_count>, if specified, takes precedence over an <if_name>, if specified at all.

4.7. Source Stream Control

4.7.1. INSERT - Insert Secondary Source.

insert

Description:

The **insert** pseudo instruction provides a means of obtaining source statements from a file other than that being used for input. The assembler transfers the text from this file and assembles it before taking the next statement from the interrupted source of statements.

There are no parameters for the **insert** pseudo instruction. The file to be used is specified when the assembler is called (see 6.) The file will be rewound before using it. Under UNIX , the preprocessor *cpp* supports the inclusion of files and some other features. *cpp* may be used with the assembler (see 6.). Its usage is described in

Kernighan B. W.; Ritchie D. M.:
The C Programming Language.
Prentice-Hall Software Series; Chapter 4.11

4.8. Listing Control

The pseudo instructions described in this section permit extensive control of the assembly listing format.

4.8.1. LIST - Listing

list $<op_1>, <op_2>, ..., <op_n>$

$<op_i>$

Optional parameter. A list option or a list option prefixed by a minus sign. The unprefixed option selects the option; the prefixed option cancels it. Options are separated by commas and terminated by a blank. The following options are available:

dc When dc is selected, the source line of the dc pseudo instruction and its expansion are listed, otherwise only the source line will be listed.
-dc is the default.

if When if is selected, the source lines of the **ifeq, ifne, else** or **endif** pseudo instructions and the skipped source statements in the **if** range are listed, otherwise the pseudo instructions are listed, but not the skipped source statements.
-if is the default.

macro
When macro is selected, the source line of the macro reference and the fully expanded macro body are listed, otherwise only the source line of the outermost macro reference of possibly nested macro calls is listed.
-macro is the default.

xopc
When xopc is selected, the assembler will list the use of all opcodes in the cross reference list.
-xopc is the default.

xpse
When xpse is selected, the assembler will list the use of all pseudo instruction in the cross reference list.
-xpse is the default.

xreg
When xreg ist selected, the assembler will list the use of all registers in the cross reference list.
-xreg is the default.

Description:
The **list** pseudo instruction controls the content and format of the assembler listing. Use of the **list** pseudo instruction is optional. If not specified in a module, or if specified without parameters, the assembler will produce an output according to the default for each possible option.

For compatibility with the Motorola assembler the pseudo instruction **g** is also accepted.

g

Description:
The effect of the **g** pseudo instruction is identical with the effect of

list dc

4.8.2. NOLIST - Turn off Listing

nolist
nol

Description:
The **nolist** pseudo instruction suppresses the printing of the assembly listing until a **list** pseudo instruction is encountered.

4.8.3. PAGE - Top of Page.

page

Description:
The **page** pseudo instruction advances printer paper to a new page before printing. Then page headings are printed and listing continues. The **page** pseudo instruction does not appear on the program listing.

4.8.4. NOPAGE - Turn off Paging.

nopage

Description:
The **nopage** pseudo instruction suppresses paging to the output device. Page and line numbers in the cross reference and error listing will be meaningless.

4.8.5. SPC - Space Between Source Lines.

spc <count>

<count>
An absolute value.

Description:
The **spc** pseudo instruction causes the assembler to output <count> blank lines to the assembly listing. The **spc** pseudo instruction does not appear on the program listing.

4.8.6. TTL - Assembly Listing Title.

4.8.7. STTL - Assembly Listing Subtitle.

ttl '<text>'
sttl '<text>'

Description:
The **ttl** and **sttl** pseudo instruction allow the programmer to print a title and a subtitle on the top of each page of the listing. To this effect the assembler maintains internally two text strings which are set to blank at the beginning of pass one. In pass two, whenever a new page is started, these two text strings together with other information are printed in the page header. Specifying a title or subtitle merely means, that the contents of the corresponding internal text string is changed to the one specified with the **ttl** or **sttl** pseudo instruction. It does not imply an automatic start of a new page. The first specified title is in addition kept in a third internal text string and is copied into the title text string at the start of pass two. Neither the **ttl** nor the **sttl** pseudo instruction are listed in the assembly listing.

4.8.8. FAIL - Generate an Error Message.

fail

Description:
An error message is printed on the assembly listing.

4.9. Object Code Control

The pseudo instructions **blong, bshort, flong** or **fshort** allow the programmer to influence the assembler's choice whenever forward references or relative symbols are encountered, be it for direct addresses or relative branching instructions.

4.9.1. BLONG - Use Two-Word Branch.

4.9.2. BSHORT - Use One-Word Branch.

blong
bshort

Description:
The two pseudo instructions **blong** and **bshort** allow the programmer to influence the assembler whenever it is assembling a forward reference. By default the assembler will use the two-word instruction form allowing a larger relative address range. After a **bshort** pseudo instruction the assembler will generate the one-word relative branch instruction, unless the suffix .l a **blong** pseudo instruction forces the two-word relative branch instruction to be generated, unless the suffix .s has been appended to that branch instruction.

4.9.3. FLONG - Force Direct Long Address.

4.9.4. FSHORT - Force Direct Short Address.

flong
fshort

Description:
The two pseudo instructions **flong** and **fshort** allow the programmer to influence the assembler whenever it is assembling an direct address the label of which contains a forward reference. By default the assembler will use the long direct address form. After an **fshort** pseudo instruction the assembler will generate the direct short address form. The occurrence of a **flong** pseudo instruction forces the direct long address form to be generated.

☛ The selected option, long or short direct addresses, is only valid until the next occurrence of a **flong**, **fshort** or segment control pseudo instruction.

4.9.5. NOOBJ - Suppress a.out Output.

noobj

Description:
The pseudo instruction **noobj** suppresses the generation of a a.out module.

4.10. Cross Reference Control

The pseudo instructions allow the programmer to select whether a cross reference listing shall be built up and printed at the end of the assembler listing. Default is **xrefon**.

xrefon

Description:

A cross reference listing is built up and printed.

xrefoff

Description:

A cross reference listing is suppressed.

5. MACRO OPERATIONS

A macro definition is a sequence of source statements that are saved and then assembled whenever needed through a macro call. A macro call consists of the occurrence of the macro name in the operation field of a statement. It usually includes parameters to be substituted for formal parameters in the macro code sequence so that code generated can vary with each assembly of the definition.

Use of a macro requires two steps, definition of the macro, and calling of the definition.

A definition consists of three parts: heading, body, and terminator.

heading — A macro definition is headed by a **macro** pseudo instruction stating the name of the macro. The heading optionally includes a **local** pseudo instruction identifying symbols local to the definition.

body — The body begins with the first statement in a definition that is not a **local** pseudo instruction or a comment line. The body consists of a series of symbolic instructions. All instructions other than **end** or another **macro** definition are legal within a definition. The assembler recognizes substitutable arguments in all fields of the source line. The macro argument ~0 however can only be used in the operation field for referring to the data size subparameter in an opcode or pseudo instruction. The arguments ~1 through ~9 can appear anywhere in a source line. Ten is the maximum number of arguments that can be handled by any macro definition. Macro calls may be nested up to ten levels deep.

terminator — An **endm** pseudo instruction terminates a macro definition.

5.1. ENDM - End Macro Definition.

endm

An **endm** pseudo instruction terminates the macro definition.

5.2. LOCAL - Local Symbols.

local <sym$_1$>, <sym$_2$>,..., <sym$_n$>

<sym$_i$>

List of local symbols. Symbols must be separated by commas. A blank terminates the list. The maximum number of local symbols is 10.

Description:

The **local** pseudo instruction, which lists symbols local to the definition optionally follows the **macro** pseudo instruction.

A symbol in the list is considered local to the macro; that is, it is known only within the macro definition. On each expansion of the macro, the assembler creates a new symbol for each local symbol and substitutes it for each occurrence of the local symbol in the definition. Thus invented symbols replace **local** named symbols wherever they appear in a macro definition in a manner similar to the way substitutable parameters are replaced.

5.3. MACRO - Macro Heading.

<m_name> **macro**

<m_name>
A mandatory symbol that defines the name of the macro.

Description:
A **macro** pseudo instruction tells the assembler to place the instructions forming the body of the macro in a table of macro definitions for assembly upon call, and to place the macro name in the symbol table.

5.4. MACRO CALLS

A macro headed by a **macro** pseudo instruction can be called by an instruction in the following format:

<symbol>	**<m_name>**	$<p_1>, <p_2>, ..., <p_n>$
<symbol>	**<m_name>.s**	$<p_1>, <p_2>, ..., <p_n>$
<symbol>	**<m_name>.b**	$<p_1>, <p_2>, ..., <p_n>$
<symbol>	**<m_name>.w**	$<p_1>, <p_2>, ..., <p_n>$
<symbol>	**<m_name>.l**	$<p_1>, <p_2>, ..., <p_n>$

<symbol>
An optional location symbol.

<m_name>
Name of a previously defined macro. The (optional) size attribute substitutes macro parameter 0 (see above).

$<p_i>$

Parameter list composed of strings of characters. Parameters are separated by commas and terminated by a blank. Two consecutive commas constitute a null parameter.

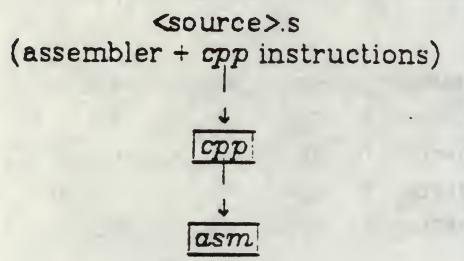
If null parameters are interspersed with non-null parameters, the correct positions must be established with commas. When the list terminates before the last possible parameter, all remaining parameters are considered null.

When the first character of a parameter is a left angle bracket (<), the assembler considers all the characters between it and the matching right angular bracket (>) as one parameter. The assembler removes the outer pair of angle brackets before substituting the enclosed character string in a line. Embedded brackets must be properly paired. A bracketed item can contain blanks and commas.

6. HOW TO USE THE ASSEMBLER

The assembler has been designed as a two pass assembler and is written in Pascal. To run the assembler two input files (INPUT - assembler source, INSERT - accessed only by the **insert** pseudo instruction), two output files (OUTPUT - assembler listing, AOUT - a.out object module) and two work files (SCRATCH, XREFFIL) must be provided.

Under UNIX the shell procedure *as* runs the preprocessor *cpp* and the assembler *asm*. One or more assembler source files may be listed as arguments. There is no provision for assignment of INSERT since insertion can be achieved more conveniently by use of *cpp*. Following UNIX convention, AOUT is written on a file named <name>.o, while the assembler sources should be named <name>.s. Listings will be displayed to <name>.lst.



Example 1:

as mysource.s
reads mysource.s
displays the listing to mysource.lst, the object module to mysource.o

Example 2:

as mysource*.s
reads mysource0.s mysource1.s mysource3.s
files the listings to mysource0.lst mysource1.lst mysource3.lst
writes the object modules to mysource0.o mysource1.o mysource3.o

Should you experience any problems or encounter errors, please contact the author.

7. APPENDIX

Symbols and related address modes and relocation information

symbol type	textsegment	datasegment
absolute	DA (r_abs)	DA (r_abs)
text	PCD16 (r_abs) DA (r_text)	DA (r_text)
data	DA (r_data)	PCD16 (r_abs) DA (r_text)
bss	DA (r_bss)	DA (r_bss)
external	DA (r_ext)	DA (r_ext)

Notation:

DA direct addressing mode

PCD16 program counter relative addressing mode (generated if requested)

Related relocation information

- r_abs (absolute symbol)
- r_text (text symbol)
- r_data (data symbol)
- r_bss (bss symbol)
- r_ext (external symbol)

List of Error Messages

0:constant too large
1:character not defined for m 68000 assembler
2:character missing or not valid for constant
3:string too long or not terminated properly
4:entry point or external symbol multiply defined
5:entry point not defined
7:symbol cannot be used as a label
8:this operation needs a label
9:this operation does not allow a label
10:symbol multiply defined
12:symbol not defined
13:initial **ifeq** or **ifne** is missing or misplaced
14:the label of an **if** should not be used here
15:**if** conditions more than maxinest levels deep
16:symbol cannot be used as an opcode
17:size specification illegal or not allowed
18:macro expansion error
19:more than maxmlocal local parameters
20:at most one **local** pseudo per macrodefinition
21:initial **macro** definition missing or misplaced
22:**macro** calls more than maxmnest levels deep
23:> expected
24:do not nest macro definitions
25:opcode/macro or pseudocode missing
26:no such cross-reference option
28:operation needs one or more operands
29:address or data register expected
30:address register expected
31:bad termination of an expression
32:an expression cannot start with this symbol
33:an operand cannot start with this symbol
34:the count must be absolute for this pseudo
35:the count must be positive for this pseudo
36:the expression must be greater lc for **org** pseudo
38:displacements are restricted to byte or word size
40:argument(s) missing in expression
41:displacement is restricted to byte size
42:type conflict between address and program counter
43:expression too complicated, use **equ** or **set** pseudo to break it down
44:expression too large for size specified
45:forward reference not allowed for this pseudo
46:both arguments must be absolute for logical operations
47:.**s** option does not allow branch to next word
48:register specification expected
49:) expected
50:) or , expected
51:separator expected
52:no size specification for this operation
53:byte size specification not allowed
54:both arguments must be absolute for shift operations
55:string expected

List of Error Messages (continued)

56:too many operands are specified for this operation
57:both arguments must be absolute for * or /
58:this address mode is illegal for the opcode
59:address mode combination illegal for opcode
60:do not write a comment on this line
61:synchronization error between pass one and two
62:too many errors this instruction line
63:**fail** generated error, consult listing
64:syntax error in register list for **movem**
65:**org** argument is of illegal type
68:no code generation in bss segment
69:list options are: dc, if, macro, xopc, xpse, xreg
70:one argument must be absolute for add operation
71:illegal operand types for sub operation
72:a.out buffers exceeded

FSCK—The UNIX File System Check Program

T. J. Kowalski

Bell Laboratories
Murray Hill, New Jersey 07974

ABSTRACT

The UNIX† File System Check Program (*fsck*) is an interactive file system check and repair program. *Fsck* uses the redundant structural information in the UNIX file system to perform several consistency checks. If an inconsistency is detected, it is reported to the operator, who may elect to fix or ignore each inconsistency. These inconsistencies result from the permanent interruption of the file system updates, which are performed every time a file is modified. *Fsck* is frequently able to repair corrupted file systems using procedures based upon the order in which UNIX honors these file system update requests.

The purpose of this document is to describe the normal updating of the file system, to discuss the possible causes of file system corruption, and to present the corrective actions implemented by *fsck*. Both the program and the interaction between the program and the operator are described.

1. INTRODUCTION

When a UNIX operating system is brought up, a consistency check of the file systems should always be performed. This precautionary measure helps to insure a reliable environment for file storage on disk. If an inconsistency is discovered, corrective action must be taken. No changes are made to any file system by *fsck* without prior operator approval.

The purpose of this memo is to dispel the mystique surrounding file system inconsistencies. It first describes the updating of the file system (the calm before the storm) and then describes file system corruption (the storm). Finally, the set of heuristically sound corrective actions used by *fsck* (the Coast Guard to the rescue) is presented.

2. UPDATE OF THE FILE SYSTEM

Every working day hundreds of files are created, modified, and removed. Every time a file is modified, the UNIX operating system performs a series of file system updates. These updates, when written on disk, yield a consistent file system. To understand what happens in the event of a permanent interruption in this sequence, it is important to understand the order in which the update requests were probably being honored. Knowing which pieces of information were probably written to the file system first, heuristic procedures can be developed to repair a corrupted file system.

There are five types of file system updates. These involve the super-block, inodes, indirect blocks, data blocks (directories and files), and free-list blocks.

2.1 Super-Block

The super-block contains information about the size of the file system, the size of the inode list, part of the free-block list, the count of free blocks, the count of free inodes, and part of the free-inode list.

The super-block of a mounted file system (the root file system is always mounted) is written to the file system whenever the file system is unmounted or a *sync* command is issued.

---

† UNIX is a Trademark of Bell Telephone Laboratories.

2.2 Inodes

An inode contains information about the type of inode (directory, data, or special), the number of directory entries linked to the inode, the list of blocks claimed by the inode, and the size of the inode.

An inode is written to the file system upon closure[1] of the file associated with the inode.

2.3 Indirect Blocks

There are three types of indirect blocks: single-indirect, double-indirect and triple-indirect. A single-indirect block contains a list of some of the block numbers claimed by an inode. Each one of the 128 entries in an indirect block is a data-block number. A double-indirect block contains a list of single-indirect block numbers. A triple-indirect block contains a list of double-indirect block numbers.

Indirect blocks are written to the file system whenever they have been modified and released[2] by the operating system.

2.4 Data Blocks

A data block may contain file information or directory entries. Each directory entry consists of a file name and an inode number.

Data blocks are written to the file system whenever they have been modified and released by the operating system.

2.5 First Free-List Block

The super-block contains the first free-list block. The free-list blocks are a list of all blocks that are not allocated to the super-block, inodes, indirect blocks, or data blocks. Each free-list block contains a count of the number of entries in this free-list block, a pointer to the next free-list block, and a partial list of free blocks in the file system.

Free-list blocks are written to the file system whenever they have been modified and released by the operating system.

3. CORRUPTION OF THE FILE SYSTEM

A file system can become corrupted in a variety of ways. The most common of these ways are improper shutdown procedures and hardware failures.

3.1 Improper System Shutdown and Startup

File systems may become corrupted when proper shutdown procedures are not observed, e.g., forgetting to *sync* the system prior to halting the CPU, physically write-protecting a mounted file system, or taking a mounted file system off-line.

File systems may become further corrupted if proper startup procedures are not observed, e.g., not checking a file system for inconsistencies, and not repairing inconsistencies. Allowing a corrupted file system to be used (and, thus, to be modified further) can be disastrous.

3.2 Hardware Failure

Any piece of hardware can fail at any time. Failures can be as subtle as a bad block on a disk pack, or as blatant as a non-functional disk-controller.

4. DETECTION AND CORRECTION OF CORRUPTION

A quiescent[3] file system may be checked for structural integrity by performing consistency

1. All in core blocks are also written to the file system upon issue of a *sync* system call.
2. More precisely, they are queued for eventual writing. Physical I/O is deferred until the buffer is needed by UNIX or a *sync* command is issued.
3. I.e., unmounted and not being written on.

checks on the redundant data intrinsic to a file system. The redundant data is either read from the file system or computed from other known values. A quiescent state is important during the checking of a file system because of the multi-pass nature of the *fsck* program.

When an inconsistency is discovered *fsck* reports the inconsistency for the operator to chose a corrective action.

Discussed in this section are how to discover inconsistencies and possible corrective actions for the super-block, the inodes, the indirect blocks, the data blocks containing directory entries, and the free-list blocks. These corrective actions can be performed interactively by the *fsck* command under control of the operator.

4.1 Super-Block

One of the most common corrupted items is the super-block. The super-block is prone to corruption because every change to the file system's blocks or inodes modifies the super-block.

The super-block and its associated parts are most often corrupted when the computer is halted and the last command involving output to the file system was not a *sync* command.

The super-block can be checked for inconsistencies involving file-system size, inode-list size, free-block list, free-block count, and the free-inode count.

4.1.1 File-System Size and Inode-List Size. The file-system size must be larger than the number of blocks used by the super-block and the number of blocks used by the list of inodes. The number of inodes must be less than 65,535. The file-system size and inode-list size are critical pieces of information to the *fsck* program. While there is no way to actually check these sizes, *fsck* can check for them being within reasonable bounds. All other checks of the file system depend on the correctness of these sizes.

4.1.2 Free-Block List. The free-block list starts in the super-block and continues through the free-list blocks of the file system. Each free-list block can be checked for a list count out of range, for block numbers out of range, and for blocks already allocated within the file system. A check is made to see that all the blocks in the file system were found.

The first free-block list is in the super-block. *Fsck* checks the list count for a value of less than zero or greater than fifty. It also checks each block number for a value of less than the first data block in the file system or greater than the last block in the file system. Then it compares each block number to a list of already allocated blocks. If the free-list block pointer is non-zero, the next free-list block is read in and the process is repeated.

When all the blocks have been accounted for, a check is made to see if the number of blocks used by the free-block list plus the number of blocks claimed by the inodes equals the total number of blocks in the file system.

If anything is wrong with the free-block list, then *fsck* may rebuild it, excluding all blocks in the list of allocated blocks.

4.1.3 Free-Block Count. The super-block contains a count of the total number of free blocks within the file system. *Fsck* compares this count to the number of blocks it found free within the file system. If they don't agree, then *fsck* may replace the count in the super-block by the actual free-block count.

4.1.4 Free-Inode Count. The super-block contains a count of the total number of free inodes within the file system. *Fsck* compares this count to the number of inodes it found free within the file system. If they don't agree, then *fsck* may replace the count in the super-block by the actual free-inode count.

4.2 Inodes

An individual inode is not as likely to be corrupted as the super-block. However, because of the great number of active inodes, there is almost as likely a chance for corruption in the inode list as in the super-block.

The list of inodes is checked sequentially starting with inode 1 (there is no inode 0) and going to the last inode in the file system. Each inode can be checked for inconsistencies involving format and type, link count, duplicate blocks, bad blocks, and inode size.

4.2.1 Format and Type. Each inode contains a mode word. This mode word describes the type and state of the inode. Inodes may be one of four types: regular inode, directory inode, special block inode, and special character inode. If an inode is not one of these types, then the inode has an illegal type. Inodes may be found in one of three states: unallocated, allocated, and neither unallocated nor allocated. This last state indicates an incorrectly formatted inode. An inode can get in this state if bad data is written into the inode list through, for example, a hardware failure. The only possible corrective action is for *fsck* is to clear the inode.

4.2.2 Link Count. Contained in each inode is a count of the total number of directory entries linked to the inode.

Fsck verifies the link count of each inode by traversing down the total directory structure, starting from the root directory, calculating an actual link count for each inode.

If the stored link count is non-zero and the actual link count is zero, it means that no directory entry appears for the inode. If the stored and actual link counts are non-zero and unequal, a directory entry may have been added or removed without the inode being updated.

If the stored link count is non-zero and the actual link count is zero, *fsck* may link the disconnected file to the *lost+found* directory. If the stored and actual link counts are non-zero and unequal, *fsck* may replace the stored link count by the actual link count.

4.2.3 Duplicate Blocks. Contained in each inode is a list or pointers to lists (indirect blocks) of all the blocks claimed by the inode.

Fsck compares each block number claimed by an inode to a list of already allocated blocks. If a block number is already claimed by another inode, the block number is added to a list of duplicate blocks. Otherwise, the list of allocated blocks is updated to include the block number. If there are any duplicate blocks, *fsck* will make a partial second pass of the inode list to find the inode of the duplicated block, because without examining the files associated with these inodes for correct content, there is not enough information available to decide which inode is corrupted and should be cleared. Most times, the inode with the earliest modify time is incorrect, and should be cleared.

This condition can occur by using a file system with blocks claimed by both the free-block list and by other parts of the file system.

If there is a large number of duplicate blocks in an inode, this may be due to an indirect block not being written to the file system.

Fsck will prompt the operator to clear both inodes.

4.2.4 Bad Blocks. Contained in each inode is a list or pointer to lists of all the blocks claimed by the inode.

Fsck checks each block number claimed by an inode for a value lower than that of the first data block, or greater than the last block in the file system. If the block number is outside this range, the block number is a bad block number.

If there is a large number of bad blocks in an inode, this may be due to an indirect block not being written to the file system.

Fsck will prompt the operator to clear both inodes.

4.2.5 Size Checks. Each inode contains a thirty-two bit (four-byte) size field. This size indicates the number of characters in the file associated with the inode. This size can be checked for inconsistencies, e.g., directory sizes that are not a multiple of sixteen characters, or the number of blocks actually used not matching that indicated by the inode size.

A directory inode within the UNIX file system has the directory bit on in the inode mode word. The directory size must be a multiple of sixteen because a directory entry contains sixteen bytes of information (two bytes for the inode number and fourteen bytes for the file or directory name).

Fsck will warn of such directory misalignment. This is only a warning because not enough information can be gathered to correct the misalignment.

A rough check of the consistency of the size field of an inode can be performed by computing from the size field the number of blocks that should be associated with the inode and comparing it to the actual number of blocks claimed by the inode.

Fsck calculates the number of blocks that there should be in an inode by dividing the number of characters in a inode by the number of characters per block (512) and rounding up. *Fsck* adds one block for each indirect block associated with the inode. If the actual number of blocks does not match the computed number of blocks, *fsck* will warn of a possible file-size error. This is only a warning because UNIX does not fill in blocks in files created in random order.

4.3 Indirect Blocks

Indirect blocks are owned by an inode. Therefore, inconsistencies in indirect blocks directly affect the inode that owns it.

Inconsistencies that can be checked are blocks already claimed by another inode and block numbers outside the range of the file system.

For a discussion of detection and correction of the inconsistencies associated with indirect blocks, apply iteratively Sections 4.2.3 and 4.2.4 to each level of indirect blocks.

4.4 Data Blocks

The two types of data blocks are plain data blocks and directory data blocks. Plain data blocks contain the information stored in a file. Directory data blocks contain directory entries. *Fsck* does not attempt to check the validity of the contents of a plain data block.

Each directory data block can be checked for inconsistencies involving directory inode numbers pointing to unallocated inodes, directory inode numbers greater than the number of inodes in the file system, incorrect directory inode numbers for "." and "..", and directories which are disconnected from the file system.

If a directory entry inode number points to an unallocated inode, then *fsck* may remove that directory entry. This condition probably occurred because the data blocks containing the directory entries were modified and written to the file system while the inode was not yet written out.

If a directory entry inode number is pointing beyond the end of the inode list, *fsck* may remove that directory entry. This condition occurs if bad data is written into a directory data block.

The directory inode number entry for "." should be the first entry in the directory data block. Its value should be equal to the inode number for the directory data block.

The directory inode number entry for ".." should be the second entry in the directory data block. Its value should be equal to the inode number for the parent of the directory entry (or the inode number of the directory data block if the directory is the root directory).

If the directory inode numbers are incorrect, *fsck* may replace them by the correct values.

Fsck checks the general connectivity of the file system. If directories are found not to be linked into the file system, *fsck* will link the directory back into the file system in the *lost+found* directory. This condition can be caused by inodes being written to the file system with the corresponding directory data blocks not being written to the file system.

4.5 Free-List Blocks

Free-list blocks are owned by the super-block. Therefore, inconsistencies in free-list blocks directly affect the super-block.

Inconsistencies that can be checked are a list count outside of range, block numbers outside of range, and blocks already associated with the file system.

For a discussion of detection and correction of the inconsistencies associated with free-list blocks see Section 4 1.2.

ACKNOWLEDGEMENT

I would like to thank Larry A. Wehr for advice that lead to the first version of *fsck* and Rick B. Brandt for adapting *fsck* to UNIX/TS.

REFERENCES

[1] Ritchie, D. M., and Thompson, K., The UNIX Time-Sharing System, *The Bell System Technical Journal* 57, 6 (July-August 1978, Part 2), pp. 1905-29.

[2] Dolotta, T. A., and Olsson, S. B. eds., *UNIX User's Manual, Edition 1.1* (January 1978).

[3] Thompson, K., UNIX Implementation, *The Bell System Technical Journal* 57, 6 (July-August 1978, Part 2), pp. 1931-46.

Appendix—FSCK ERROR CONDITIONS

1. CONVENTIONS

Fsck is a multi-pass file system check program. Each file system pass invokes a different Phase of the *fsck* program. After the initial setup, *fsck* performs successive Phases over each file system, checking blocks and sizes, path-names, connectivity, reference counts, and the free-block list (possibly rebuilding it), and performs some cleanup.

When an inconsistency is detected, *fsck* reports the error condition to the operator. If a response is required, *fsck* prints a prompt message and waits for a response. This appendix explains the meaning of each error condition, the possible responses, and the related error conditions.

The error conditions are organized by the *Phase* of the *fsck* program in which they can occur. The error conditions that may occur in more than one Phase will be discussed in initialization.

2. INITIALIZATION

Before a file system check can be performed, certain tables have to be set up and certain files opened. This section concerns itself with the opening of files and the initialization of tables. This section lists error conditions resulting from command line options, memory requests, opening of files, status of files, file system size checks, and creation of the scratch file.

C option?

C is not a legal option to *fsck*; legal options are −y, −n, −s, −S, and −t. *Fsck* terminates on this error condition. See the *fsck*(1M) manual entry for further detail.

Bad −t option

The −t option is not followed by a file name. *Fsck* terminates on this error condition. See the *fsck*(1M) manual entry for further detail.

Invalid −s argument, defaults assumed

The −s option is not suffixed by 3, 4, or blocks-per-cylinder:blocks-to-skip. *Fsck* assumes a default value of 400 blocks-per-cylinder and 9 blocks-to-skip. See the *fsck*(1M) manual entry for more details.

Incompatible options: −n and −s

It is not possible to salvage the free-block list without modifying the file system. *Fsck* terminates on this error condition. See the *fsck*(1M) manual entry for further detail.

Can't get memory

Fsck's request for memory for its virtual memory tables failed. This should never happen. *Fsck* terminates on this error condition. See a guru.

Can't open checklist file: F

The default file system checklist file F (usually */etc/checklist*) can not be opened for reading. *Fsck* terminates on this error condition. Check access modes of F.

Can't stat root

Fsck's request for statistics about the root directory "/" failed. This should never happen. *Fsck* terminates on this error condition. See a guru.

Can't stat F

Fsck's request for statistics about the file system F failed. It ignores this file system and continues checking the next file system given. Check access modes of F.

F is not a block or character device

You have given *fsck* a regular file name by mistake. It ignores this file system and continues checking the next file system given. Check file type of F.

Can't open F

The file system F can not be opened for reading. It ignores this file system and continues checking the next file system given. Check access modes of F.

Size check: fsize X isize Y

More blocks are used for the inode list Y than there are blocks in the file system X, or there are more than 65,535 inodes in the file system. It ignores this file system and continues checking the next file system given. See Section 4.1.1.

Can't create F

Fsck's request to create a scratch file F failed. It ignores this file system and continues checking the next file system given. Check access modes of F.

CAN NOT SEEK: BLK B (CONTINUE)

Fsck's request for moving to a specified block number B in the file system failed. This should never happen. See a guru.

Possible responses to the CONTINUE prompt are:

YES attempt to continue to run the file system check. Often, however the problem will persist. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If the block was part of the virtual memory buffer cache, *fsck* will terminate with the message "Fatal I/O error".

NO terminate the program.

CAN NOT READ: BLK B (CONTINUE)

Fsck's request for reading a specified block number B in the file system failed. This should never happen. See a guru.

Possible responses to the CONTINUE prompt are:

YES attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If the block was part of the virtual memory buffer cache, *fsck* will terminate with the message "Fatal I/O error".

NO terminate the program.

CAN NOT WRITE: BLK B (CONTINUE)

Fsck's request for writing a specified block number **B** in the file system failed. The disk is write-protected. See a guru.

Possible responses to the CONTINUE prompt are:

YES attempt to continue to run the file system check. Often, however, the problem will persist. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If the block was part of the virtual memory buffer cache, *fsck* will terminate with the message "Fatal I/O error".

NO terminate the program.

3. PHASE 1: CHECK BLOCKS AND SIZES

This phase concerns itself with the inode list. This section lists error conditions resulting from checking inode types, setting up the zero-link-count table, examining inode block numbers for bad or duplicate blocks, checking inode size, and checking inode format.

UNKNOWN FILE TYPE I=I (CLEAR)

The mode word of the inode I indicates that the inode is not a special character inode, special character inode, regular inode, or directory inode. See Section 4.2.1.

Possible responses to the CLEAR prompt are:

YES de-allocate inode I by zeroing its contents. This will always invoke the UNALLOCATED error condition in Phase 2 for each directory entry pointing to this inode.

NO ignore this error condition.

LINK COUNT TABLE OVERFLOW (CONTINUE)

An internal table for *fsck* containing allocated inodes with a link count of zero has no more room. Recompile *fsck* with a larger value of MAXLNCNT.

Possible responses to the CONTINUE prompt are:

YES continue with the program. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If another allocated inode with a zero link count is found, this error condition is repeated.

NO terminate the program.

B BAD I=I

Inode I contains block number B with a number lower than the number of the first data block in the file system or greater than the number of the last block in the file system. This error condition may invoke the EXCESSIVE BAD BLKS error condition in Phase 1 if inode I has too many block numbers outside the file system range. This error condition will always invoke the BAD/DUP error condition in Phase 2 and Phase 4. See Section 4.2.4.

EXCESSIVE BAD BLKS I=I (CONTINUE)

There is more than a tolerable number (usually 10) of blocks with a number lower than the number of the first data block in the file system or greater than the number of last block in the file system associated with inode I. See Section 4.2.4.

Possible responses to the CONTINUE prompt are:

YES ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system.

NO terminate the program.

B DUP I=I

Inode I contains block number B which is already claimed by another inode. This error condition may invoke the EXCESSIVE DUP BLKS error condition in Phase 1 if inode I has too many block numbers claimed by other inodes. This error condition will always invoke Phase 1b and the BAD/DUP error condition in Phase 2 and Phase 4. See Section 4.2.3.

EXCESSIVE DUP BLKS I=I (CONTINUE)

There is more than a tolerable number (usually 10) of blocks claimed by other inodes. See Section 4.2.3.

Possible responses to the CONTINUE prompt are:

YES ignore the rest of the blocks in this inode and continue checking with the next inode in the file system. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system.

NO terminate the program.

DUP TABLE OVERFLOW (CONTINUE)

An internal table in *fsck* containing duplicate block numbers has no more room. Recompile *fsck* with a larger value of DUPTBLSIZE.

Possible responses to the CONTINUE prompt are:

YES continue with the program. This error condition will not allow a complete check of the file system. A second run of *fsck* should be made to re-check this file system. If another duplicate block is found, this error condition will repeat.

NO terminate the program.

POSSIBLE FILE SIZE ERROR I=I

The inode I size does not match the actual number of blocks used by the inode. This is only a warning. See Section 4.2.5.

DIRECTORY MISALIGNED I=I

The size of a directory inode is not a multiple of the size of a directory entry (usually 16). This is only a warning. See Section 4.2.5.

PARTIALLY ALLOCATED INODE I=I (CLEAR)

Inode I is neither allocated nor unallocated. See Section 4.2.1.

Possible responses to the CLEAR prompt are:

YES de-allocate inode I by zeroing its contents.

NO ignore this error condition.

4. PHASE 1B: RESCAN FOR MORE DUPS

When a duplicate block is found in the file system, the file system is rescanned to find the inode which previously claimed that block. This section lists the error condition when the duplicate block is found.

B DUP I=I

Inode I contains block number B which is already claimed by another inode. This error condition will always invoke the BAD/DUP error condition in Phase 2. You can determine which inodes have overlapping blocks by examining this error condition and the DUP error condition in Phase 1. See Section 4.2.3.

5. PHASE 2: CHECK PATH-NAMES

This phase concerns itself with removing directory entries pointing to error conditioned inodes from Phase 1 and Phase 1b. This section lists error conditions resulting from root inode mode and status, directory inode pointers in range, and directory entries pointing to bad inodes.

ROOT INODE UNALLOCATED. TERMINATING.

The root inode (usually inode number 2) has no allocate mode bits. This should never happen. The program will terminate. See Section 4.2.1.

ROOT INODE NOT DIRECTORY (FIX)

The root inode (usually inode number 2) is not directory inode type. See Section 4.2.1.

Possible responses to the FIX prompt are:

YES	replace the root inode's type to be a directory. If the root inode's data blocks are not directory blocks, a VERY large number of error conditions will be produced.
NO	terminate the program.

DUPS/BAD IN ROOT INODE (CONTINUE)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks in the root inode (usually inode number 2) for the file system. See Section 4.2.3 and 4.2.4.

Possible responses to the CONTINUE prompt are:

YES	ignore the DUPS/BAD error condition in the root inode and attempt to continue to run the file system check. If the root inode is not correct, then this may result in a large number of other error conditions.
NO	terminate the program.

I OUT OF RANGE I=I NAME=F (REMOVE)

A directory entry F has an inode number I which is greater than the end of the inode list. See Section 4.4.

Possible responses to the REMOVE prompt are:

YES	the directory entry F is removed.
NO	ignore this error condition.

UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F (REMOVE)

A directory entry F has an inode I without allocate mode bits. The owner O, mode M, size S, modify time T, and file name F are printed. See Section 4.4.

Possible responses to the REMOVE prompt are:

YES the directory entry F is removed.
NO ignore this error condition.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory entry F, directory inode I. The owner O, mode M, size S, modify time T, and directory name F are printed. See Section 4.2.3 and 4.2.4.

Possible responses to the REMOVE prompt are:

YES the directory entry F is removed.
NO ignore this error condition.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T FILE=F (REMOVE)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory entry F, inode I. The owner O, mode M, size S, modify time T, and file name F are printed. See Section 4.2.3 and 4.2.4.

Possible responses to the REMOVE prompt are:

YES the directory entry F is removed.
NO ignore this error condition.

6. PHASE 3: CHECK CONNECTIVITY

This phase concerns itself with the directory connectivity seen in Phase 2. This section lists error conditions resulting from unreferenced directories, and missing or full *lost+found* directories.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)

The directory inode I was not connected to a directory entry when the file system was traversed. The owner O, mode M, size S, and modify time T of directory inode I are printed. See Section 4.4 and 4.2.2.

Possible responses to the RECONNECT prompt are:

YES reconnect directory inode I to the file system in the directory for lost files (usually *lost+found*). This may invoke the *lost+found* error condition in Phase 3 if there are problems connecting directory inode I to *lost+found*. This may also invoke the CONNECTED error condition in Phase 3 if the link was successful.
NO ignore this error condition. This will always invoke the UNREF error condition in Phase 4.

SORRY. NO lost+found DIRECTORY

There is no *lost+found* directory in the root directory of the file system; *fsck* ignores the request to link a directory in *lost+found*. This will always invoke the UNREF error condition in Phase 4. Check access modes of *lost+found*. See *fsck*(1M) manual entry for further detail.

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the *lost+found* directory in the root directory of the file system; *fsck* ignores the request to link a directory in *lost+found*. This will always invoke the UNREF error condition in Phase 4. Clean out unnecessary entries in *lost+found* or make *lost+found* larger. See *fsck*(1M) manual entry for further detail.

DIR I=I1 CONNECTED. PARENT WAS I=I2

This is an advisory message indicating a directory inode I1 was successfully connected to the *lost+found* directory. The parent inode I2 of the directory inode I1 is replaced by the inode number of the *lost+found* directory. See Section 4.4 and 4.2.2.

7. PHASE 4: CHECK REFERENCE COUNTS

This phase concerns itself with the link count information seen in Phase 2 and Phase 3. This section lists error conditions resulting from unreferenced files, missing or full *lost+found* directory, incorrect link counts for files, directories, or special files, unreferenced files and directories, bad and duplicate blocks in files and directories, and incorrect total free-inode counts.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT)

Inode I was not connected to a directory entry when the file system was traversed. The owner O, mode M, size S, and modify time T of inode I are printed. See Section 4.2.2.

Possible responses to the RECONNECT prompt are:

YES reconnect inode I to the file system in the directory for lost files (usually *lost+found*). This may invoke the *lost+found* error condition in Phase 4 if there are problems connecting inode I to *lost+found*.

NO ignore this error condition. This will always invoke the CLEAR error condition in Phase 4.

SORRY. NO lost+found DIRECTORY

There is no *lost+found* directory in the root directory of the file system; *fsck* ignores the request to link a file in *lost+found*. This will always invoke the CLEAR error condition in Phase 4. Check access modes of *lost+found*.

SORRY. NO SPACE IN lost+found DIRECTORY

There is no space to add another entry to the *lost+found* directory in the root directory of the file system; *fsck* ignores the request to link a file in *lost+found*. This will always invoke the CLEAR error condition in Phase 4. Check size and contents of *lost+found*.

(CLEAR)

The inode mentioned in the immediately previous error condition can not be reconnected. See Section 4.2.2.

Possible responses to the CLEAR prompt are:

YES de-allocate the inode mentioned in the immediately previous error condition by zeroing its contents.

NO ignore this error condition.

LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for inode I which is a file, is X but should be Y. The owner O, mode M, size S, and modify time T are printed. See Section 4.2.2.

Possible responses to the ADJUST prompt are:

YES	replace the link count of file inode I with Y.
NO	ignore this error condition.

LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for inode I which is a directory, is X but should be Y. The owner O, mode M, size S, and modify time T of directory inode I are printed. See Section 4.2.2.

Possible responses to the ADJUST prompt are:

YES	replace the link count of directory inode I with Y.
NO	ignore this error condition.

LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X SHOULD BE Y (ADJUST)

The link count for F inode I is X but should be Y. The name F, owner O, mode M, size S, and modify time T are printed. See Section 4.2.2.

Possible responses to the ADJUST prompt are:

YES	replace the link count of inode I with Y.
NO	ignore this error condition.

UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Inode I which is a file, was not connected to a directory entry when the file system was traversed. The owner O, mode M, size S, and modify time T of inode I are printed. See Section 4.2.2 and 4.4.

Possible responses to the CLEAR prompt are:

YES	de-allocate inode I by zeroing its contents.
NO	ignore this error condition.

UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Inode I which is a directory, was not connected to a directory entry when the file system was traversed. The owner O, mode M, size S, and modify time T of inode I are printed. See Section 4.2.2 and 4.4.

Possible responses to the CLEAR prompt are:

YES	de-allocate inode I by zeroing its contents.
NO	ignore this error condition.

BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with file inode I. The owner O, mode M, size S, and modify time T of inode I are printed. See Section 4.2.3 and 4.2.4.

Possible responses to the CLEAR prompt are:

YES de-allocate inode I by zeroing its contents.
NO ignore this error condition.

BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR)

Phase 1 or Phase 1b have found duplicate blocks or bad blocks associated with directory inode I. The owner O, mode M, size S, and modify time T of inode I are printed. See Section 4.2.3 and 4.2.4.

Possible responses to the CLEAR prompt are:

YES de-allocate inode I by zeroing its contents.
NO ignore this error condition.

FREE INODE COUNT WRONG IN SUPERBLK (FIX)

The actual count of the free inodes does not match the count in the super-block of the file system. See Section 4.1.4.

Possible responses to the FIX prompt are:

YES replace the count in the super-block by the actual count.
NO ignore this error condition.

8. PHASE 5: CHECK FREE LIST

This phase concerns itself with the free-block list. This section lists error conditions resulting from bad blocks in the free-block list, bad free-blocks count, duplicate blocks in the free-block list, unused blocks from the file system not in the free-block list, and the total free-block count incorrect.

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE)

The free-block list contains more than a tolerable number (usually 10) of blocks with a value less than the first data block in the file system or greater than the last block in the file system. See Section 4.1.2 and 4.2.4.

Possible responses to the CONTINUE prompt are:

YES ignore the rest of the free-block list and continue the execution of *fsck*. This error condition will always invoke the BAD BLKS IN FREE LIST error condition in Phase 5.
NO terminate the program.

EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE)

The free-block list contains more than a tolerable number (usually 10) of blocks claimed by inodes or earlier parts of the free-block list. See Section 4.1.2 and 4.2.3.

Possible responses to the CONTINUE prompt are:

YES ignore the rest of the free-block list and continue the execution of *fsck*. This error condition will always invoke the DUP BLKS IN FREE LIST error condition in Phase 5.

NO terminate the program.

BAD FREEBLK COUNT

The count of free blocks in a free-list block is greater than 50 or less than zero. This error condition will always invoke the BAD FREE LIST condition in Phase 5. See Section 4.1.2.

X BAD BLKS IN FREE LIST

X blocks in the free-block list have a block number lower than the first data block in the file system or greater than the last block in the file system. This error condition will always invoke the BAD FREE LIST condition in Phase 5. See Section 4.1.2 and 4.2.4.

X DUP BLKS IN FREE LIST

X blocks claimed by inodes or earlier parts of the free-list block were found in the free-block list. This error condition will always invoke the BAD FREE LIST condition in Phase 5. See Section 4.1.2 and 4.2.3.

X BLK(S) MISSING

X blocks unused by the file system were not found in the free-block list. This error condition will always invoke the BAD FREE LIST condition in Phase 5. See Section 4.1.2.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)

The actual count of free blocks does not match the count in the super-block of the file system. See Section 4.1.3.

Possible responses to the FIX prompt are:

YES replace the count in the super-block by the actual count.

NO ignore this error condition.

BAD FREE LIST (SALVAGE)

Phase 5 has found bad blocks in the free-block list, duplicate blocks in the free-block list, or blocks missing from the file system. See Section 4.1.2, 4.2.3, and 4.2.4.

Possible responses to the SALVAGE prompt are:

YES replace the actual free-block list with a new free-block list. The new free-block list will be ordered to reduce time spent by the disk waiting for the disk to rotate into position.

NO ignore this error condition.

9. PHASE 6: SALVAGE FREE LIST

This phase concerns itself with the free-block list reconstruction. This section lists error conditions resulting from the blocks-to-skip and blocks-per-cylinder values.

Default free-block list spacing assumed

This is an advisory message indicating the blocks-to-skip is greater than the blocks-per-cylinder, the blocks-to-skip is less than one, the blocks-per-cylinder is less than one, or the blocks-per-cylinder is greater than 500. The default values of 9 blocks-to-skip and 400 blocks-per-cylinder are used. See the *fsck*(1M) manual entry for further detail.

10. CLEANUP

Once a file system has been checked, a few cleanup functions are performed. This section lists advisory messages about the file system and modify status of the file system.

X files Y blocks Z free

This is an advisory message indicating that the file system checked contained X files using Y blocks leaving Z blocks free in the file system.

******* BOOT UNIX (NO SYNC!) *******

This is an advisory message indicating that a mounted file system or the root file system has been modified by *fsck*. If UNIX is not rebooted immediately, the work done by *fsck* may be undone by the in-core copies of tables UNIX keeps.

******* FILE SYSTEM WAS MODIFIED *******

This is an advisory message indicating that the current file system was modified by *fsck*. If this file system is mounted or is the current root file system, *fsck* should be halted and UNIX rebooted. If UNIX is not rebooted immediately, the work done by *fsck* may be undone by the in-core copies of tables UNIX keeps.

May 1979

INDEX OF MESSAGES
(Alphabetically within each section)

INITIALIZATION

PHASE 1: CHECK BLOCKS AND SIZES

PHASE 1B: RESCAN FOR MORE DUPS

PHASE 2: CHECK PATH-NAMES

PHASE 3: CHECK CONNECTIVITY

PHASE 4: CHECK REFERENCE COUNTS

PHASE 5: CHECK FREE LIST

PHASE 6: SALVAGE FREE LIST

CLEANUP

Berkeley Font Catalog

October 1980

Introduction

This catalog gives samples of the various fonts available at Berkeley using vtroff on our Versatec and Varian. We have them working 4 pages across in a 36 inch Versatec, and rotated 90 degrees on a Benson-Varian 11 inch plotter. The same software should be adaptable to an 11 inch Versatec, and in fact is running at several other sites, however, not having one here, it isn't part of this distribution. Such a driver is available from Tom Ferrin at UCSF.

To use these fonts:

(1) Hershey. This is the default font. The Hershey font is currently the *only* complete font, with all 16 point sizes and all the special characters troff knows about. To get it, use vtroff directly. To illustrate this with the –ms macro package:

vtroff –ms paper.nr

(2) Fonts with roman, italic, and bold, such as nonie. You can load all three fonts with, for example:

vtroff –F nonie –ms paper.nr

To get just one of these fonts, use (3) below, appending .r, .i, or .b to the font name to specify which font you want mounted, e.g., to get italics in delegate,

vtroff –2 delegate.i –ms paper.nr

(3) To get a font without a complete set, choose which font (1, 2, or 3) you want replaced by the chosen font. For example, to use bocklin as though it were bold, since font 3 is bold, use:

vtroff –3 bocklin –ms paper.nr

To switch between fonts in troff, use

.ft 3

to switch to font 3, for example, or use

\f3word\f1

to switch within a line. For more information see the Nroff/Troff Users Manual.

Special note: *troff* thinks it is talking to a CAT phototypesetter. Thus, it does all sorts of strange things, such as enforcing restrictions like 7.54 inches maximum width, 4 fonts, a certain 16 point sizes, proportional spacing by point size, etc.

In particular, the following glyphs will *always* be taken from the special font, no matter what font you are using at the time:

©, #, ", ´, `, <, >, \, {, }, ~, ^, and _

This may explain what are otherwise surprising results in some of the subsequent pages.

In addition, the following Greek letters have been decreed by *troff* as looking so much like their Roman counterparts that the Roman version (font 1) is always printed, no matter what font is mounted on font 1 at the time:

A, B, E, Z, H, I, K, M, N, O, P, T, X.

(See table II in the back of the Nroff/Troff Users's Manual for details about what glyphs are in each font and how to generate the special glyphs.)

Font Layout Positions

Code	Normal		Special		Code	Normal	Special	
000					100			
001	fi	\(fi	∎	\(if	101	A	A	\(*A
002	fl	\(fl	⊃	\(ip	102	B	B	\(*B
003	ff	\(ff	∝	\(pt	103	C	Γ	\(*G
004	–	\-	∎	\(rh	104	D	Δ	\(*D
005	_	\(ru	∪	\(cu	105	E	E	\(*E
006	—	\(em	∩	\(rn	106	F	Z	\(*Z
007	•	\(bu	∈	\(bs	107	G	H	\(*Y
010	•	\(sq	±	\(+-	110	H	Θ	\(*H
011		\(fI	≤	\(<=	111	I	I	\(*I
012		\(fL	≥	\(>=	112	J	K	\(*K
013	°	\(de	√	\(sr	113	K	Λ	\(*L
014	†	\(dg	∽	\(ts	114	L	M	\(*M
015	'	\(fm	∫	\(is	115	M	N	\(*N
016	●	\(co	∕	\(sl	116	N	Ξ	\(*C
017	●	\(rg	|	\(bv	117	O	O	\(*O
020	¢	\(ct	|	\(lf	120	P	Π	\(*P
021	¼	\(14	|	\(rf	121	Q	P	\(*R
022	½	\(12	⌈	\(lc	122	R	Σ	\(*S
023	¾	\(34	|	\(rc	123	S	T	\(*T
024			⌊	\(lt	124	T	T	\(*U
025			⌋	\(lb	125	U	Φ	\(*F
026			⌉	\(rt	126	V	X	\(*X
027			⌋	\(rb	127	W	Ψ	\(*Q
030			{	\(lk	130	X	Ω	\(*W
031			}	\(rk	131	Y	‡	\(dd
032			⊂	\(sb	132	Z	—	\(br
033			⊃	\(sp	133	[	⊂	\(ib
034			∩	\(ca	134		∕	\(ib
035			∘	\(no	135	]	○	\(ci
036			∎	\(lh	136		~	
037			∈	\(mo	137	_	˙	\(*.
040	space				140			
041	!				141	a	α	\(*a
042			"		142	b	β	\(*b
043			#		143	c	γ	\(*g
044	$				144	d	δ	\(*d
045	%				145	e	ε	\(*e
046	&				146	f	ζ	\(*z
047			'		147	g	η	\(*y
050	(		∀	\(gr	150	h	θ	\(*h
051	)				151	i	ι	\(*i
052	*		×	\(mu	152	j	κ	\(*k
053	+		+	\(pl	153	k	λ	\(*l
054	,				154	l	μ	\(*m
055	-		–	\(mi	155	m	ν	\(*n
056	.				156	n	ξ	\(*c
057	/		÷	\(di	157	o	ο	\(*o
060	0		≡	\(==	160	p	π	\(*p
061	1		≡	\(~=	161	q	ρ	\(*r
062	2		~	\(ap	162	r	σ	\(*s
063	3		≠	\(!=	163	s	τ	\(*t
064	4		←	\(<-	164	t	υ	\(*u
065	5		→	\(->	165	u	φ	\(*f
066	6		↑	\(ua	166	v	χ	\(*x
067	7		↓	\(da	167	w	ψ	\(*q
070	8		§	\(sc	170	x	ω	\(*w
071	9		°	\(**	171	y	∂	\(pd
072	:				172	z	ϕ	\(em
073	;				173		←	
074			<		174	|	—	\(ul
075			=		175		~	
076			>		176			
077	?				177			

APL FONT, 10 POINT ONLY

A∝B⊥C∩D⌊E∈ F_G∇H∆I⍳J⍤K' L⌷M⍕N⍴O○P Q?R⍟S⌈T~U↓ V∪W X⊃Y↑Z⊂ 01234 56789*

(" # $ = × ≥ ∨ ∧ ≤ ≠ + ⋆⋆ { } { } ⍲ ~ _ \ ∃ ⍝ → < ÷ / \ . > , <

*! → (⅍ → = & → × ' → ' (→ ∨) → ∧ : → ≤ * → ≠ - → ÷ = → ⋆⋆ [→ {] → } | → ∃ ' → → ; → < + → + ? → *

Baskerville font, roman, ibold, italic, 12 point only (Called "basker" on line.)

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ‘ ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ‘ ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ‘ ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Bocklin font, 14 and 28 point only.

14 point

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz
01234 56789

'' ( ) : - = [ ] ' ; / ? . .

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality: since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

28 point (No punctuation except period.)

ABCDE FGHIJ KLMNO PQRST
UVWXYZ abcde fghij klmno pqrst
uvwxyz 01234 56789 .

If time be of all things the most precious wasting time must be as Poor Richard says the greatest prodigality since as he elsewhere tells us lost time is never found again and what we call time enough always proves little enough Let us then up and be doing and doing to the purpose so by diligence shall we do more with less perplexity.

Bodoni font, roman, **bold**, *italic*, 10 point only.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Chess, 18 point only

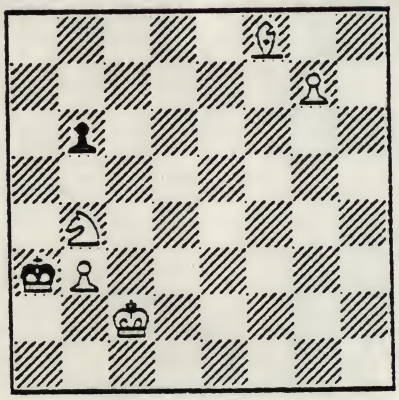
Note: Our attempt at compatibility with Stanford was only 99% successful. If you use a blank space to indicate an empty white square it will come out narrow due to the stupidity of troff. Either include the line

```
.cs ch 36
```

to put yourself in constant spacing mode or else use zero instead of space. You should also set the vertical spacing to 18 points.

```
.nf
.ft ch
.cs ch 36
.ps 18
.vs 18
HTTTTTTTTX
VOZOZOAOZF
VZOZOZOOOF
VOoOZOZOZF
VZOZOZOZOF
VOMOZOZOZF
VjPZOZOZOF
VOZKZOZOZF
VZOZOZOZOF
WUUUUUUUUG
.sp
.ft P
.ps 8
.cs P
```

p P
o O
b B
a A
n N
m M
r R
s S
q Q
l L
k K
j J
U T
F V
G W
X H
0 Z



White mates in three moves.

Clarendon, 14 and 18 point roman only. From SAIL (Paul Martin & Andy Moorer)

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

" # $ % × ' ( ) : - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

" # $ % × ' ( ) : - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Computer Modern fonts, roman, *italic*, and bold (by Don Knuth) 6, 7, 8, 9, 10, 11, 12 point. (Available as cm)

Note that the cm fonts are intended for TEX and don't fare so well with troff. The spacing is not proportional by point size, and hence only one point size can be tuned to be nicely spaced. We have tuned the 10 point size, but the 8 point looks somewhat cramped.

Some of the punctuation is missing in some of the fonts. Knuth also uses a nonstandard notion of ASCII, and hence some glyphs are available only with special symbols such as \(12. Others cannot be accessed at all.

Knuth's fonts somewhat larger than normal, since he intends the output to be reduced before printing. Since troff has a limitation of 754 inches width on output, this is not practical. Hence, the original fonts have been relabelled with the point size they are closest to without reduction. Some fonts (6 point bold, 7 point roman, 8 point italic and bold, 9 point bold, and 11 point italic) which would have otherwise been missing were generated by shrinking the next larger point size of the same style. (This goes against the idea of metafont, but we use the tools we have.)

10 Point Roman

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234
56789 ! " # $ % & ' ( ) * + - / : ; < = > ? @ [\ ] ^ _ ` { | } ~ ¡ ¿ ß æ œ ø Æ Œ Ø Ł ª º

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality since, as he elsewhere tells us, lost time is never found again and what we call time enough, always proves little enough Let us then up and be doing, and doing to the purpose so by diligence shall we do more with less perplexity.

10 Point Italic

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234
*56789 ! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ @ ` ; + / ? . > , < ' ' Σ, ¨, ¨, Ξ, Υ, Φ, Π, η, θ, ι, Δ, Θ, Λ, Ψ, Ω, α, β, γ, ζ ξ δ*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

10 Point Bold

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234
56789 ! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ @ ` ; + / ? . > , < ' ' Σ, ¨, ¨, Ξ, Υ, Φ, Π, ˘, ¯, ¨, Δ, Θ, Λ, Ψ, Ω, ı, ȷ, ¸, ˛, ˚

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

6 Point Roman, Bold, and *Italic*
7 Point Roman, Bold, and *Italic*
8 Point Roman, Bold, and *Italic*
9 Point Roman, Bold, and *Italic*
10 Point Roman, Bold, and *Italic*
11 Point Roman, Bold, and *Italic*
12 Point Roman, Bold, and *Italic*

Countdown (22 point, upper case letters only.) From SAIL (Paul Martin)

ABCDE FGHIJ KLMNO PQRST UVWXYZ

COUNTDOWN HAS NO INTEGERS TO COUNT DOWN WITH BUT IT COMPENSATES BY BEING UGLY AND ILLEGIBLE

Cyrillic, 12 point only

ЖЦъЗ абде фгхи клмно прст увйз

ф тиме бе оф алл тхингс тхе мост преноус астинг тиме муст бе ас оор ихард сайс тхе греатест продигалитй сине ас хе елсехере теллс ус лост тиме ис невер фоунд агаин анд хат е алл тиме еноугх алайс провес литтле еноугх ет ус тхен уп анд бе доинг анд доинг то тхе пурпосе со бй дклигене схалл е до море итх лесс перплеитй

W→Ж X→Ц Y→ъ Z→З a→а b→б d→д e→е f→ф g→г h→х i→и k→к l→л m→м n→н o→о p→п r→р s→с t→т u→у v→в y→й z→з

Delegate, roman, *italic*, and **bold**, 12 point only

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ↑ ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ↑ ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Fix fixed width font, 6, 9, 10, 12, 14 point

6 point

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] _ | ' ; + / ? . > . <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

9 point

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

10 point

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

12 point

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity. ...

14 point

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Gacham, roman, bold, *italic*, 18 point only
The gacham font is almost indistinguishable from the fix font. In fact, it has been pointed out that our gacham roman and bold fonts really *are* fix. Sigh. They are included anyway for convenience.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789
*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789
! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Greek, 10 point only

This font provides an alternative to the Greek characters on the standard special font.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxy

ΑΒΧΔΕ ΦΓΗΙφ ΚΛΜΝΟ ΠΘΡΣΤ ΤΩΞΨΖ αβχδε φγηιφ κλμνο πθρστ ψωξψζ

Ιφ τιμε βε οφ αλλ τηινγσ τηε μοστ πρεχιουσ ωαστινγ τιμε μυστ βε ασ Ποορ Ριχηαρδ σαψσ τηε γρεατεστ προδιγαλιτψ σινχε ασ ηε ελσεωηερε τελλσ υσ λοστ τιμε ισ νεπερ φουνδ αγαιν ανδ ωηατ ωε χαλλ τιμε ενουγη αλωαψσ προυεσ λιττλε ενουγη Λετ υσ τηεν βε δοινγ ανδ δοινγ το τηε πυρποσε σο βψ διλιγενχε σηαλλ ωε δο μορε ωιτη λεσσ περπλεξιτψ

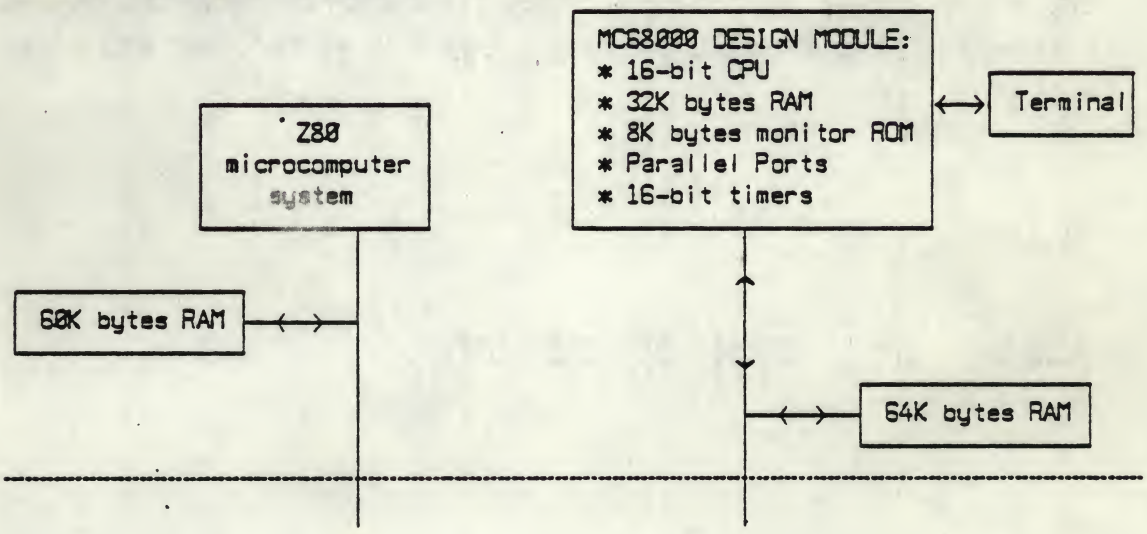
The h19 font includes a subset of the h19's graphic charactor set, plus a few logical extensions to allow forms and diagrams to be drawn. The charactors are the same as the h19's graphic interpratation set.

` a b c d e f s t u v m n h i k l

| ─ ┼ ┐ ┘ └ ┌ ┬ ┤ ┴ ├ ┄ ┼ → ← ↓ ↑

The charactors are designed to overlap.

Example of usage for diagrams:



Hebrew, 16, 24, and 36 point only

16 point

עדכבא עיהגפ ונמלב טשרקפ ויחתוו בֿא ךף ובלפ שפ יחתו 01234 56789

! " # ף צ . ( ) : - [ ] { } ^ ~ _ \ | @ " ; ץ ? . > . <

עהט שאה טי .שטיטסטידערפ רנגפורפ הלפ עסיהס טנעללעסחע נא טי טטלפ תודאהש עהט
אואחא טא ! .טקיפ ורפ ף ! וללו וא טי וף תאט .עלבאדאערנו טשטלא גנידעב פי עגאטנאדא
ף ןיב טבלא טבאנ.לב

24 point

דסבא הג ע נמלב טשרקפ ויחתו ם

& → צ :

רנ רפ ר ס הס טנ לל סח נא ש טנ ת דאהש הט
טש טלא גנ ב גאטנאודא הט שאה ט .שנ טס ד רפ
לבאדא רנ .

36 point (rather ragged).

ויחתו טשרקפ נמלב ע הג דסבא

צ .

10 point Hershey

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789 !, $, %, &, ', (,), :, *, -, [,], ', ;, /, ?, .

\(em → —, -- → -, \- → –, \(bu → •, \(sq → ▪, \(ru → _ \(14 → ¼, \(12 → ½, \(34 → ¾, \(fi → fi, \(fl → fl, \(ff → ff, \(Fi → ffi, \(Fl → ffl, \(de → °, \(dg → †, \(fm → ′, \(ct → ¢\(rg → ®\(co → ©
When you flex your fingers in a coffin, it can baffle a giraffe.

*ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789 !, $, %, &, ', (,), :, *, -, [,], ', ;, /, ?, .*

\(em → —, -- → -, \- → –, \(bu → •, \(sq → ▪, \(ru → _ \(14 → ¼\(12 → ½\(34 → ¾\(fi → *fi*, \(fl → *fl*, \(ff → *ff*, \(Fi → *ffi*, \(Fl → *ffl*, \(de → °, \(dg → †, \(fm → ′, \(ct → ¢\(rg → ®\(co → ©
When you flex your fingers in a coffin, it can baffle a giraffe.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789 !, $, %, &; ', (,), :, *, -, [,], ', ;, /, ?, .

\(em → —, -- → -, \- → –, \(bu → •, \(sq → ▪, \(ru → _, \(14 → ¼\(12 → ½\(34 → ¾\(fi → **fi**, \(fl → **fl**, \(ff → **ff**, \(Fi → **ffi**, \(Fl → **ffl**, \(de → °, \(dg → †, \(fm → ′, \(ct → ¢\(rg → ®\(co → ©
When you flex your fingers in a coffin, it can baffle a giraffe.
From special font: " # = { } ^ ~ _ \ | @ ` ' + > <

Special characters: \(pl → +, \(mi → −, \(eq → =, \(** → *, \(sc → §, \(aa → ´, \(ga → `, \(ul → _, \(sl → /, \(*a → α, \(*b → β, \(*g → γ, \(*d → δ, \(*e → ε, \(*z → ζ, \(*y → η, \(*h → θ, \(*i → ι, \(*k → κ, \(*l → λ, \(*m → μ, \(*n → ν, \(*c → ξ, \(*o → ο, \(*p → π, \(*r → ρ, \(*s → σ, \(ts → ς, \(*t → τ, \(*u → υ, \(*f → φ, \(*x → χ, \(*q → ψ, \(*w → ω, \(*A → Α, \(*B → Β, \(*G → Γ, \(*D → Δ, \(*E → Ε, \(*Z → Ζ, \(*Y → Η, \(*H → Θ, \(*I → Ι, \(*K → Κ, \(*L → Λ, \(*M → Μ, \(*N → Ν, \(*C → Ξ, \(*O → Ο, \(*P → Π, \(*R → Ρ, \(*S → Σ, \(*T → Τ, \(*U → Υ, \(*F → Φ, \(*X → Χ, \(*Q → Ψ, \(*W → Ω, \(sr → √, \(rn → ¯, \(>= → ≥, \(<= → ≤, \(== → ≡, \(~= → ≅, \(ap → ~, \(!= → ≠, \(-> → →, \(<- → ←, \(ua → ↑, \(da → ↓, \(mu → ×, \(di → ÷, \(+- → ±, \(cu → ∪, \(ca → ∩, \(sb → ⊂, \(sp → ⊃, \(ib → ⊆, \(ip → ⊇, \(if → ∞, \(pd → ∂, \(gr → ∇, \(no → -, \(is → ∫, \(pt → ∝, \(eq → =, \(no → -, \(br → ¦, \(dd → ‡, \(rh → ▬\(lh → ◀, \(bs → \, \(or → |, \(ci → ○, \(lt → ⎫, \(lb → ⎩, \(rt → ⎧, \(rb → ⎭, \(lk → ⎨, \(rk → ⎬, \(bv → ⎪, \(lf → ⎣, \(rf → ⎦, \(lc → ⎡, \(rc → ⎤

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again: and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

This is an *example* of a sample in **various** fonts.

Hershey font. This is the default font for vtroff. Roman, *Italic* and **Bold** in 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 28, and 36 point. The following examples are 10 point.

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

6 point Roman, **Bold**, and *Italic*.
7 point Roman, **Bold**, and *Italic*.
8 point Roman, **Bold**, and *Italic*.
9 point Roman, **Bold**, and *Italic*.
10 point Roman, **Bold**, and *Italic*.
11 point Roman, **Bold**, and *Italic*.
12 point Roman, **Bold**, and *Italic*.
14 point Roman, Bold, and *Italic*.
16 point Roman, Bold, and *Italic*.
18 point Roman, Bold, and *Italic*.
20 point Roman, Bold, and *Italic*.
22 point Roman, Bold, and *Italic*.
24 point Roman, Bold, and *Italic*.
28 point Roman, Bold, and *Italic*.
36 point Roman, Bold, and *Italic*.

Meteor, roman, bold, *italic*, 8, 10, 12 point, no 12 point italic.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Microgramma font, 10 point only

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Mona font, 24 point only

ABCDE FGHIJ KLMNO PQRST UVWXYZ
abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ ¢ & ' ( ) : - { } ^ ~ _ \ @ ; ? . > , <

Philadelphia is the most pecksniffian of American cities, and thus probably leads the world.

- H. L. Mencken

Nonie, roman, **bold**, *italic*, 8, 10, 12 point

8 point
ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

10 point
ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

12 point
ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : ª - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : ª - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

! " # $ % & ' ( ) : ª - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Old English, 8, 14, and 18 point only. (This font is called "oldenglish" on line.)

8 point

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

" # ' :- { } ^ ~ _ \ @ ; .>.<

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality;since, as he elsewhere tells us, lost time is never found again;and what we call time enough, always proves little enough:Let us then up and be doing, and doing to the purpose;so by diligence shall we do more with less perplexity.

14 point

ABCDE FGHIJ KLMNO PQRST UVWXY abcde fghij klmno pqrst uvwxyz 01234 56789

" # : { } ^ ~ _ \ @ ; .>.<

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality;since, as he elsewhere tells us, lost time is never found again;and what we call time enough, always proves little enough:Let us then up and be doing, and doing to the purpose;so by diligence shall we do more with less perplexity.

18 point

ABCDE FGHIJ KLMNO PQRST UVWXYZ
abcde fghij klmno pqrst uvwxyz 01234 56789

" # '{} - { } ^ ~ - \ ?.>,<

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality since, as he elsewhere tells us, lost time is nver found again and what we call time enough, always proves little enough and I think I'm wasting time typing all this stuff

PIP FONT, 16 POINT ONLY, NO LOWER CASE

ABCDE FGHIJ KLMNO PQRST UVWXYZ 01234 56789

! " # ' ( ) : - { } ^ ~ _ \ @ ‘ ; ? . > , <

IT COULD PROBABLY BE SHOWN BY FACTS AND FIGURES THAT THERE IS NO DISTINCTLY NATIVE AMERICAN CRIMINAL CLASS EXCEPT CONGRESS.

– MARK TWAIN

Playbill font, 18 point only

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 0123

! " # $ % & ' ( ) : * - + [ ] { } ^ ~ _ \ @ ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

Script, 18 point only. This font appears to be almost identical to the "Coronet" font from SAIL, except that the period and one other glyph of Coronet are missing a row, and Coronet is supposed to be 16 point. (They are both really the same size.)

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789

" # : { } ^ ~ _ \ @ ; . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

SHADOW, 16 POINT ONLY, NO LOWER CASE

ABCDE FGHIJ KLMNO PQRST UVWXYZ 01234 56789

!"# ' : [ ] { } ^ ~ _ \ @ ' ; + . > , <

THE SHADOW FONT IS AN EXCELLENT CHOICE FOR PROFOUND PREDICTIONS. IT HAS THE ADVANTAGE OF BEING ALMOST UNREADABLE.

++ S H A D O W ++

SIGN, 22 POINT ONLY

ABCDE FGHIJ KLMNO PQRST
UVWXYZ ➢➣ 0123456789

!"# ' :* - = { } ^ ~ _ @ ; / . > , <

THIS FONT WAS INVENTED BY A DRAFTSMAN WHO HAD LOST HIS FRENCH CURVE. ➢SO IT GOES ➣

LOWER CASE L IS ➢, LOWER CASE R IS ➣.

Stare hershey font. This font is identical to the hershey font except that the point sizes are one point smaller, and the width tables are those used for the real typesetter. Hence, this font is useful when previewing documents that are to be sent to a typesetter to make sure the spacing, paging, and so on is right. There are Roman, *Italic* and **Bold** in 8, 9, 10, 11, 12, 14, and 16 point. The following examples are 10 point.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9

*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <*

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9

! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ` ; + / ? . > , <

If time be of all things the most precious, wasting time must be, as Poor Richard says, the greatest prodigality; since, as he elsewhere tells us, lost time is never found again; and what we call time enough, always proves little enough: Let us then up and be doing, and doing to the purpose; so by diligence shall we do more with less perplexity.

8 point Roman, **Bold**, and *Italic*.
9 point Roman, **Bold**, and *Italic*.
10 point Roman, **Bold**, and *Italic*.
11 point Roman, **Bold**, and *Italic*.
12 point Roman, **Bold**, and *Italic*.
14 point Roman, **Bold**, and *Italic*.
16 point Roman, **Bold**, and *Italic*.

Times fonts, roman, *italic*, and **bold**. 10 point only.
These fonts showed up in a directory labelled "timesroman" along with three other fonts which turned out to be nonie, meteor, and news gothic. They are probably not really times fonts, but seem to be pretty close. Notice the top of the "2" for a clear difference from a real Times Roman font.

It is our desire to have a real, digitized version of the times fonts from the phototypesetter. We eventually plan to do this. At that point, the times font will probably replace the hershey font as the default. Such a Times font is already available from Johns Hopkins University for a fee, but we couldn't redistribute it, so we plan do digitize them ourselves.

10 Point
ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789
! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <
', ', —, -, -, -, •, □, , ¼, ½, ¾, fi, fl, ff, ffi, ffl, °, †, ', ¢, ∞

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789
*! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <*
', ', —, -, -, -, •, □, , ¼, ½, ¾, fi, fl, ff, ffi, ffl, °, †, ', ¢, ∞

ABCDE FGHIJ KLMNO PQRST UVWXYZ abcde fghij klmno pqrst uvwxyz 01234 56789
! " # $ % & ' ( ) : * - = [ ] { } ^ ~ _ \ | @ ' ; + / ? . > , <
', ', —, -, -, -, •, □, , ¼, ½, ¾, fi, fl, ff, ffi, ffl, °, †, ', ¢, ∞